Succinct Arguments for BatchQMA and Friends under 6 Rounds*
(OR: How to prove soundness of succinct quantum arguments without rewinding)

Rishab Goyal †
UW-Madison

Aditya Jain ‡
UW-Madison

Shashwatha Mitra GB §
UW-Madison

Abstract

We study the problem of minimizing round complexity in the context of succinct classical argument systems for quantum computation. All prior works either require at least 8 rounds of interaction between the quantum prover and classical verifier, or rely on the idealized quantum random oracle model (QROM). We design:

1. A 4-round public-coin (except for the first message) argument system for batchQMA languages. Our results come in two flavors:
   (a) Under the post-quantum hardness of functional encryption and learning with errors (LWE), we achieve optimal communication complexity (i.e., all message sizes are independent of batch size).
   (b) Under the post-quantum hardness of LWE, we achieve optimal communication complexity except for the verifier's first message.
2. A 6-round private-coin argument system for *monotone policy* batchQMA languages, under the post-quantum hardness of LWE. The communication complexity is independent of the batch size as well as the monotone circuit size.

One of our main technical contributions is a new approach to prove soundness *without rewinding cheating provers*. We bring the notion of straight-line partial extractability to argument systems for quantum computation. All previous works crucially relied on rewinding cheating provers, thus needed *"state-preserving" succinct arguments of knowledge* (AoKs) for **NP** to prove soundness.

*This is the full version of the CRYPTO 2025 paper.

†Email: rishab@cs.wisc.edu.
‡Email: ajain222@wisc.edu.
§Email: ghantebanupr@wisc.edu.

Contents

1 Introduction **1**

2 Technical Overview **3**

3 Preliminaries **13**
- 3.1 Hash Family with Succinct Local Openings . . . 14
- 3.2 Predicate Extractable Hash functions with Tags for Bit-fixing predicates . . . 14
- 3.3 Somewhere Extractable BARG (seBARG) for NP . . . 17
- 3.4 Fully Homomorphic Encryption . . . 18
- 3.5 Puncturable Pseudo-random Functions (PPRF) . . . 19
- 3.6 Indistinguishability Obfuscation (iO) . . . 19
- 3.7 Differing Inputs Obfuscation (diO) . . . 20
- 3.8 State Preserving Arguments of Knowledge . . . 21
- 3.9 Classical Verification for QMA . . . 21

4 Interactive Batch Arguments for QMA **27**
- 4.1 Construction . . . 28
- 4.2 Security Analysis . . . 30

5 Public Coin Batch Arguments for QMA **34**
- 5.1 FHE based AND Tree Construction . . . 34
- 5.2 Construction . . . 36
- 5.3 Security Analysis . . . 39

6 Making 4-round Batch Arguments for QMA fully-succinct **43**
- 6.1 Construction . . . 43
- 6.2 Security Analysis . . . 44

7 Monotone Policy Batch Arguments for QMA **51**
- 7.1 Construction . . . 52
- 7.2 Security Analysis . . . 56

8 Public Coin Monotone Policy Batch Arguments for QMA **62**
- 8.1 Construction . . . 63
- 8.2 Security Analysis . . . 64

9 Non-Interactive Batch Arguments in QROM **67**

10 Public Verifiability using Differing inputs iO **69**
- 10.1 Construction . . . 69
- 10.2 Security Analysis . . . 70

A Public Coin Succinct Arguments for QMA **79**
- A.1 Quantum Knowledge extraction from Classical PoK systems . . . 79
- A.2 Succinct Argument systems for QMA . . . 80

B Proof of Lemma 7.23 **82**

1 Introduction

Succinct verification of classical computation is one of the most fundamental concepts in computer science. It has numerous connections across theoretical computer science and is extremely relevant for many practical applications. The first succinct argument was designed by Kilian [Kil92], relying on probabilistically checkable proof (PCP) systems [BFL91, ALM+98, AS98] and collision-resistant hash functions (CRHFs). Kilian's argument system is an interactive protocol between a prover and a verifier to prove the validity of any **NP** statement. It requires 4 rounds of interaction, and the communication complexity grows only polylogarithmically with the size of the **NP** witness.

Starting with Kilian's seminal work, a large body of work (see [Mic00, Gro10, BCCT12, BCCT13, SW14, Gro16, BCI+13, BCC+17, BHK17b, KPY19a, BKK+18, JKKZ21, CJJ22, KLVW23, BBK+23, NWW23, JKLV24] and references therein) has investigated computationally sound proof systems (commonly called arguments [BCC88]) that have low communication complexity. A large number of these design succinct *non-interactive* arguments (SNARGs) for **NP** and its numerous sub-classes (e.g., **P**, batch**NP**) in many different setup models (e.g., CRS, random oracles, designated verifier) under the hardness of varying cryptographic assumptions. Such arguments allow a "weak" classical client to delegate a classical computation to a "powerful" server, such that the server can convince the client of correctly performing computation, yet the client can catch a cheating server that runs in classical polynomial time. Although we do not yet know how to design SNARGs for all **NP** from standard falsifiable assumptions [Nao03] in the plain model, and this remains a very active and exciting area of research.

This work. We study the problem of succinct *classical* verification of *quantum* computation [CCY20, BKL+22]. The goal is to design succinct arguments that can be used by a single *quantum* server to prove correctness of quantum computation to a *classical* client. We require the verifier to be convinced in polylog-time with only polylog-bits of classical communication, and not get fooled by a cheating server. Prior works [CCY20, BKL+22, GKNV24, MNZ24] have successfully constructed *designated-verifier* SNARGs for QMA (and sub-classes such as batchQMA) with a classical verifier by relying on the post-quantum hardness of the learning with errors (LWE) problem [Reg05] in the quantum random oracle model (QROM). As in the classical setting, one of the research goals in this area is to design succinct arguments for QMA and its sub-classes in the standard model, without relying on QROM.

Unfortunately, we do not yet have non-interactive succint arguments for *even sub-classes*[1] *of* QMA in the plain model! Even more embarrassingly, we do not even know how to design *interactive* succinct arguments for any sub-class of QMA with the same round complexity as Kilian's original 4-round construction. In other words, a tantalizing open question is:

Can we design 4-round succinct arguments with classical verification for QMA *(or its sub-classes) from standard falsifiable assumptions in the plain model?*

All prior works [CCY20, BKL+22, GKNV24, MNZ24] need at least 8 rounds of interaction between the quantum prover and classical verifier. We believe that answering the above question will be instrumental in designing *round-optimal* succinct arguments for quantum computation with classical verification. Similar approach has been very successful in the classical setting towards designing *round-optimal* succinct arguments for various sub-classes of **NP**.

[1] Throughout, by sub-classes of QMA we mean those that cannot be classically simulated.

Our results. We answer the above question affirmatively for the class of batchQMA computations, which corresponds to batched (parallel) verification of many different QMA computations, where the proof size does not grow with the batch size. We also show how to reduce round complexity for the class of *monotone policy* batchQMA computations, which is a quantum analogue of monotone policy batch**NP** languages [BBK+23], where the prover could prove an arbitrary monotone circuit over the validity of a batch of QMA computations. Informally, we summarize our results as follows:

1. Assuming post-quantum hardness of LWE, there is a *4-round public-coin* (except for the first message) argument system for batchQMA. Here, the communication complexity (apart from the verifier's first message[2]) is independent of the batch size.

2. Assuming post-quantum hardness of LWE and functional encryption[3], there is a *4-round public-coin* (except for the first message) succinct argument system for batchQMA. Here the total communication complexity is independent of the batch size.

3. Assuming post-quantum hardness of LWE, there is a *6-round private-coin* argument system for *monotone policy* batchQMA. Here, the communication complexity is independent of the batch size and the monotone circuit size.

Our interactive arguments significantly depart from all prior works [BKL+22, GKNV24, MNZ24] that crucially relied on *"state-preserving" arguments of knowledge* (AoKs) for **NP** [CMSZ22, LMS22] for proving soundness of their argument systems. At a high level, prior works required a composable extraction property due to which they had to rely on a mild variant of Kilian's 4-round protocol to ensure any cheating prover can be rewound without significantly disturbing its state.

The core insight behind our argument systems is that recursive/composable extraction is not necessary to prove soundness for succinct argument systems for quantum computation. That is, we can completely avoid using state-preserving AoKs to design succinct argument systems for many non-trivial sub-classes of QMA. As we briefly discuss in Appendix A, prior works are stuck at 8 rounds due to their usage of Kilian-style (state-preserving) AoKs for **NP** (and it is unclear whether existing protocols could be improved any further while correctly proving soundness). By avoiding the use of such AoKs, we are able to go under the 8 round mark.

As a side contribution, we believe that our approach significantly reduces the complexity of correlation intractable (CI) hash functions [CGH04] that would be sufficient to make our succinct arguments non-interactive by applying the Fiat-Shamir [FS86] methodology. Prior works relied on significantly stronger CI hashes, as they need to make the Kilian-style AoKs for **NP** also non-interactive which is known to be a very hard task [GW11]. We remark that similar approaches (of reducing the the complexity of necessary CI hashes) have been successful in the classical literature [CJJ22]. Thus, we believe our simplified template for interactive succinct arguments is an important step towards designing succinct non-interactive arguments for non-trivial classes of QMA in the standard model.

We also formalize a natural technique to make these arguments publicly-verifiable under the non-standard assumption of differing inputs obfuscation [BCP14, ABG+13, Zha23]. Next, we provide a high level technical overview of our main ideas.

[2]Here the first round message is reusable, thus one can view the first round message as a common reference string (CRS). Thus, this could also be viewed as a 3-round succinct argument system for batchQMA with a long CRS.

[3]In the main body, we prove it under post-quantum hardness of indistinguishability obfuscation (iO) [BGI+01], but only need iO for circuits with log-length inputs. By relying on known transformations [AJ15, BV15, AJS15] from FE [BSW11] to iO, we can relax our assumption to only polynomial hardness of FE.

2 Technical Overview

Succinct classical verification of QMA computations is usually formalized as an interactive protocol between a quantum QPT prover exchanging *classical* messages with a classical PPT verifier. Crucially, the verifier does not have any quantum computing capabilities. Completeness states that an honest verifier must always accept whenever a prover honestly runs the protocol given valid QMA witness(es).[4] Succinctness requires that the combined size of all messages between the prover and verifier should not grow with the size of the QMA witness.

Soundness for succinct arguments states that no QPT adversary should be able to efficiently compute (classical) responses to an honest verifier during the protocol execution such that the verifier declares the corresponding instance to be valid, when actually the instance is invalid. This can be formalized in different ways, and two most common approaches are non-adaptive and adaptive soundness. The difference between the two can be understood by looking at the information about the cheating instance that must be declared prior to the start of the protocol, i.e. before the verifier's first message. Non-adaptive soundness requires the instance to be fully declared before the protocol starts, while adaptive soundness puts no such constraints. In this work, we mostly stick to non-adaptive soundness, however we also prove some of our succinct arguments to be semi-adaptively sound [CJJ22], which is known to be strictly stronger than non-adaptive soundness.

The focus of our work is on two popular sub-classes, the class of batched computations [RRR16] and *monotone policy* batched computations [BBK+23]. Batched quantum computations, denoted by batchQMA languages, is defined w.r.t. any QMA language $\mathcal{L}$. Where each instance in the batched language $\mathcal{L}^{\otimes k}$ consists of a batch of (say) k instances $x_1, \ldots, x_k$, and the instance is in the language iff every $x_i \in \mathcal{L}$ for $i \in [k]$. Monotone policy batched quantum computations, denoted by monotone-batchQMA, are a powerful generalization of regular batchQMA, where each language is associated with a function f that can be implemented as a monotone (classical) circuit. An instance containing $x_1, \ldots, x_k$ is said to be in the language iff there exists a sequence of bits $b_1, \ldots, b_k$ such that $f(b_1, \ldots, b_k) = 1$ *and* $b_i = 1$ only if $x_i \in \mathcal{L}$. That is, it generalizes the predicate defined over the satisfiability of the underlying batch of instances to monotone circuits. Note that batchQMA languages are a specialization of monotone-batchQMA, where f simply corresponds to the AND predicate (i.e., $f_{\mathsf{batchQMA}}(b_1, \ldots, b_k) = \wedge_i b_i$). Colloquially, succinct arguments for batchQMA are also referred to as batch arguments (BARGs) for QMA, and similarly, succinct arguments for monotone-batchQMA are also referred to as monotone policy BARGs for QMA.

BARGs and monotone policy BARGs for **NP** have received tremendous attention over the last several years. Currently, we have multiple approaches to design such succinct argument systems in the literature. In terms of (plausible) post-quantum secure succinct arguments, [CJJ22] designed BARGs for **NP** as well as [BBK+23] designed monotone policy BARGs for **NP** under the LWE assumption. By relying on post-quantum hardness of LWE, these give us post-quantum non-interactive succinct arguments for both classes. A natural question that we study is:

Can we design succinct arguments, with classical verifiers, for their quantum analogues?

To the best of our knowledge, designing succinct arguments for batchQMA or monotone-batchQMA, with classical verifier, in the plain model from standard assumptions does not face strong barriers [GW11].

[4] We inherit the need for polynomially-many copies of the witness from prior works [FHM18, CCY20, BKL+22].

What do we currently know? Succinct interactive arguments for QMA in the plain model were first constructed in [BKL+22] using (post-quantum) indistinguishability obfuscation ($i\mathcal{O}$) and the LWE problem. Moreover, they designed interactive BARGs for QMA under the hardness of LWE problem (i.e., without relying on $i\mathcal{O}$). They could also make their protocols public-coin *except the first message*. By this we mean that the verifier's messages (except the first round message) are simply random coins with no inherent structure. They also describe how to transform their protocols into succinct non-interactive arguments in the Quantum Random Oracle model (QROM). In two very recent works [GKNV24, MNZ24], the dependence on post-quantum $i\mathcal{O}$ was removed. This led to the first construction for succinct interactive arguments for QMA, without $i\mathcal{O}$, from the post-quantum hardness of LWE. Although these works relied on diverging techniques, they directly build on the protocols designed in [BKL+22]. All these protocols required 8 or more rounds of interaction. (We discuss the 8-round limitation further in Appendix A.)

In this work, we go beyond the 8-round barrier in the plain model. To explain our main ideas, we provide a brief recap of the original design from [BKL+22].

Recapping succinct interactive arguments. A typical classical verification protocol for quantum computation uses two key building blocks: the FHM protocol [FHM18] and Mahadev's measurement protocol [Mah18]. The FHM protocol is a QMA verification protocol with a quantum verifier. The verifier measures a quantum state presented to it by the prover and runs a classical algorithm to complete the verification. Mahadev [Mah18] constructed a classical verification protocol, where a classical verifier can retrieve measurement results from a (possibly malicious) quantum prover. This is an interactive protocol which requires the prover to submit 'classical' commitments to a quantum state. Following this, the verifier queries the prover and verifies the response using the commitment information it already possesses. Consequently, these two protocols can be combined together to build a QMA verification protocol with a classical verifier. Later on, [ACGH20] further improved this template and made the setup (selection of basis for measurement) instance-independent by showing that the same soundness guarantees can be reached by having a constant number of additional parallel repetitions and using random basis choices.

At a high level, the main idea in [BKL+22] was to design a *verifier-succinct* interactive protocols with classical verification. By verifier-succinct, it means that the verification protocol is such that the verifier's messages are fully succinct. That is, the prover's responses could be very large, but the verifier only sends short messages. [BKL+22] showed that by combining ideas from FHM as well as Mahadev's measurement protocol, one can design 'verifier-succinct' interactive succinct arguments for QMA by using $i\mathcal{O}$ to compress the verifier's messages. Moreover, for the case of interactive batch arguments for QMA, they showed that $i\mathcal{O}$ was not necessary and simply reusing the verifier's message for every instance in the underlying batch is sufficient.

They [BKL+22] showed that any *verifier-succinct* interactive protocols can be generically boosted to a fully-succinct protocol by relying on succinct arguments of knowledge (AoK) for classical computation. In a few words, the idea to make the prover's messages short was for the prover to send a short hashed digest instead of the full message and run an AoK to establish the existence of the pre-image (satisfying the desired predicate). Quite clearly, to prove security, one needs the AoKs to be fully extractable. Since the argument system is succinct, thus it is necessary to rewind a cheating prover. However, given that a cheating prover is a QPT adversary, each rewinding could significantly damage its state, thereby rendering extraction impossible. However, [BKL+22] proved that by relying on recent results in *state-preserving* AoKs [CMSZ22, LMS22], the above paradigm can be successfully used to design and prove the soundness of classical verification protocols for

QMA. We note that recent works follow a similar template, thus we refer the reader to [GKNV24] for a detailed discussion.

The canonical BARG protocol. Below, we sketch the BARG protocol from [BKL+22] obtained by instantiating the tools as described above. Afterwards, we will discuss our improvement to reduce the round complexity of BARGs. Consider a QMA language $\mathcal{L}$ and a batch of instances $x = (x_1, \ldots, x_k)$. As in all prior works, we consider that a prover holds polynomially many copies of each quantum witness $|\phi_j\rangle \in \mathcal{R}_{\mathcal{L}}(x_j)$.

1. The verifier samples fresh randomness r, computes the $(\mathsf{pk}, \mathsf{sk})$ pair for the Mahadev protocol, and sends pk to the prover.
2. For each $j \in [k]$, the prover computes the proof state $|\psi_j\rangle$ from the FHM protocol and then commits to it using the Mahadev protocol to get $(y_j, |\mathsf{st}_j\rangle)$. Here it reuses pk for every instance x_j. It then computes a short digest $\hat{y} = \mathcal{H}(y_1, \ldots, y_k)$ and sends $\hat{y}$ to the verifier. This is followed by an AoK for the statement: $\exists w_1, \ldots, w_k$ s.t. $\hat{y} = \mathcal{H}(w_1, \ldots, w_k)$.
3. The verifier samples a random challenge bit $c \in \{0, 1\}$ and sends it to the prover.
4. For each $j \in [k]$, the prover computes the opening using the Mahadev protocol to get z_j. Subsequently, the prover computes another short digest $\hat{z} = \mathcal{H}(z_1, \ldots, z_k)$ and sends $\hat{z}$ to the verifier. This is followed by another AoK for the statement: $\exists w_1, \ldots, w_k$ s.t. $\hat{z} = \mathcal{H}(w_1, \ldots, w_k)$.
5. Finally, the verifier sends the randomness r that it used to sample $(\mathsf{pk}, \mathsf{sk})$ in the beginning.
6. Let us denote the joint FHM and Mahadev instance verification predicate as $\mathsf{VerifyInstance}$. The prover runs this for each instance and initiates a final AoK for the statement: $\exists w_1^{(1)}, \ldots, w_k^{(1)}$ and $\exists w_1^{(2)}, \ldots, w_k^{(2)}$ s.t. $\hat{y} = \mathcal{H}(w_1^{(1)}, \ldots, w_k^{(1)})$ and $\hat{z} = \mathcal{H}(w_1^{(2)}, \ldots, w_k^{(2)})$, and the joint verification procedure accepts, i.e., $\mathsf{VerifyInstance}(x_j, w_j^{(1)}, w_j^{(2)}, c; r) = 1$ for all $j \in [k]$.

An immediate observation is that the verifier only performs classic operations, and the message size in each round does not scale with the batch size, k. Moreover, as already mentioned, soundness relies on the *full extractability* of all three state-preserving AoKs. Basically, through repeated rewindings, we can extract all the AoK witnesses. This means we can extract $y_1, \ldots, y_k$ and $z_1, \ldots, z_k$, and then later, we could use this to reduce the security to that of FHM and Mahadev protocols.

It is quite evident that the full extraction property is extremely crucial, but it also turns out to be a major source of inefficiency. This is because each AoK requires at least 2 rounds of interaction. (Technically, it has four rounds, but the first round is reusable and can be merged with the current first round of the verifier, and the second round of Kilian can be merged with the prover's last message it sends before starting the AoK.) Therefore, the above protocol needs at least 10 rounds of interaction between the prover and the verifier. (Please refer to Appendix A for more discussion.)

A natural thought and why it fails. A natural strategy would be to simply design and rely on a state-preserving non-interactive AoK? Clearly, this would immediately bring down the round complexity to 6 rounds by removing 4 unnecessary rounds of interaction due to the interactive AoKs. Unfortunately, non-interactive AoK for **NP** is a very challenging problem, and faces very strong barriers [GW11, CGKS23]. Thus, it is unclear whether the above strategy of using non-interactive AoK can be successfully instantiated.

Although full extraction in a single round appears to be a significant roadblock, our starting observation is that partial extraction might still be possible. Moreover, this might not even need

rewinding the prover, thus the notion of state-preserving arguments could potentially be totally bypassed.

Our observation draws inspiration from the classical setting, where similar issues were previously encountered and bypassed from standard cryptographic assumptions. Specifically, in the context of designing BARGs for **NP**, Choudhuri, Jain, and Jin [CJJ21, CJJ22] introduced the concept of *somewhere extractable* batch arguments (seBARGs) for **NP**. They designed such batch arguments from standard falsifiable assumptions, and demonstrated that known barriers surrounding *adaptive full extraction* [BHK17b] no longer apply for somewhere extractability.

Briefly, somewhere extractability for BARGs states that, from an accepting proof π, we can extract a single witness $w_{i^\star}$ for the $i^\star$-th instance $x_{i^\star}$ in the corresponding batch of instance. Here $i^\star$ is a special extraction index that can be programmed in the CRS of the BARG protocol, and the CRS does not leak any information about $i^\star$. A special property of somewhere extraction is that it can be performed in a straightline fashion, i.e. without rewinding a cheating prover. By relying on such a somewhere extractability (and index hiding), [CJJ21, CJJ22] and numerous follow-ups have designed BARGs for **NP** with semi-adaptive soundness. Semi-adaptive soundness is a notion that lies between non-adaptive and adaptive soundness, wherein a cheating prover must only commit an index i at the start of the protocol such that $x_i \notin \mathcal{L}$. That is, rather than requiring the adversary to commit to the entire batch of instances at the beginning (like in the non-adaptive case), this provides more flexibility to the attacker thereby providing a stronger soundness guarantee.

Partial extraction suffices. Our first idea is a rather simple observation. That is, full extraction is not necessary for designing BARGs for QMA. Basically, we can replace each AoK in the canonical BARG protocol with an appropriate seBARG. This does not affect completeness since we only need succinct classical arguments for batch**NP**-style languages in the canonical protocol. Thus, doing so would ensure that we can bring down the number of rounds to 6 as well as we can still prove semi-adaptive soundness of the resulting protocol. The idea behind the soundness proof is that we can simply extract the $i^\star$-th witness from every seBARG proof *without rewinding*, and all such witnesses combined together can be used as a *classical pseudo-witness* proving QMA satisfiability of the $i^\star$-th instance $x_{i^\star}$. By a classical pseudo-witness, we mean a classical proof in the sense of an appropriate combination of the FHM and Mahadev measurement protocol.

Consequently, this gives us a 6-round *private-coin* interactive BARG for QMA with semi-adaptive soundness. We provide more details in Section 4. We remark that it is an interesting question whether we could prove adaptive security of our protocol under slightly stronger assumptions. In the classical setting, recent works [BWW24, Goy24] partially address this problem by showing that the standard complexity leveraging technique can be used to prove adaptive soundness under sub-exponential hardness assumptions. It is unclear whether similar ideas could be used in the quantum setting, and we leave further exploration as an interesting open problem.

We want to highlight that the above BARG protocol still needs 6 rounds of interaction and is private-coin. Our intuition is that if we can make this protocol public-coin, then that could be a good step towards reducing the number of messages. Moreover, we know that for applying the Fiat-Shamir transformation, we anyways need the protocol to be public-coin. And, given how successful Fiat-Shamir transformation has been in the classical setting, it is natural to expect a similar strategy could be very influential in the quantum setting too.

We remark that many prior works [CCY20, BKL+22] relied on the same strategy, which was they utilized a folklore generic template to transform their private-coin protocol into a public-coin protocol in QROM, by leveraging post-quantum fully homomorphic encryption (FHE) [Gen09,

BV14]. We draw ideas from such a folklore private-to-public-coin paradigm to make our protocol public-coin and, along the way, further reduce its round complexity.

Getting BARGs down to 4-rounds and making them public-coin. The critical insight behind the transformation is to use FHE to encrypt the verifier's (secret) randomness at the start of the protocol and send the encrypted randomness to the prover. Given the FHE ciphertext, the prover can homomorphically perform all computations over the FHE ciphertext, and send back the encrypted output to the verifier who can decrypt and check for correctness. Clearly, this way the protocol's round complexity goes down as the verifier no longer needs to wait until the 5-th round to send the randomness r for the Mahadev protocol.

In a bit more detail, consider any succinct private-coin protocol where the verifier's per-round computation can be split into two parts – a *deterministic* computation that depends on the verifier's secret key; and a public-coin computation that is independent of the secret key. Note that the canonical BARG protocol fits such an abstraction. In such a scenario, the verifier can delegate the key-dependent deterministic computation to the prover. This is done by providing the prover a homomorphic encryption of the secret key so that the prover can run the deterministic computation homomorphically. The remainder of the protocol proceeds identically, except the prover and verifier also interact in another AoK so that the verifier can check that the homomorphic computation was performed correctly. The verifier finally decrypts the output FHE ciphertext to verify that it encrypts 1, to denote success of verification.

In the above template, it is essential that we use an AoK to prove that the homomorphic computation has been performed correctly over the FHE ciphertext, encrypting the verifier's key. This is because without the full extraction property of AoKs, it is unclear how could we catch a cheating prover. To emphasize the technical hurdle, let us simply use seBARGs instead of AoKs similar to how we used them earlier.

Attempt 1: Using seBARGs + FHE for making it public-coin. First of all, we mention that unlike the private-coin protocol, the AoK that is needed for making succinct arguments public coin is not batch**NP**-friendly. This is because the prover must: first, homomorphically evaluate the VerifyInstance circuit for the j-th instance on the FHE ciphertext encrypting verifier's secret randomness r for every $j \in [k]$; and second, it must homomorphically compute an AND gate over all these ciphertexts to preserve succinctness and ensure that all individual ciphertexts were encrypting 1 (denoting successful verification).

Now the first part is clearly batch**NP**-friendly, but second part is not necessarily batch**NP**-friendly directly. Due to the fact that it is joint computation over all individually computed ciphertexts. While this appears to be a major issue, we can solve it by combining the entire batch of k FHE ciphertexts in a tree-based fashion. That is, at each level of the tree we simply take two FHE ciphertexts from the previous layer and homomorphically add them. In other words, we can design an FHE based AND-tree that accumulates the validity of all instances in the batch. And, the special feature about this AND-tree is that, similar to Merkle hash trees, we can efficiently prove that the resulting ciphertext at the root of the tree can be recomputed by just combining ciphertexts along a Merkle-tree-style opening path in the AND-tree. The point is such an implementation, based on FHE, turns the second part of the computation into a batch**NP**-friendly computation.

We describe the construction a bit more formally below.

1. The prover commits to the witnesses, creates a digest of the commitments, and sends an seBARG proof to the verifier to prove that it has created commitments to the witnesses and

the digest contains a well-formed hash of all the commitments.

2. If the seBARG proof accepts, the verifier samples a random bit c and sends it to the prover.
3. The prover runs the verification circuit under FHE using the encrypted secret key. It then constructs an AND-tree of ciphertexts, which has a Merkle tree style opening for each instance. The prover sends an seBARG proof that shows that the AND tree was constructed correctly and the ciphertext corresponding to the root node of the AND tree.
4. The verifier accepts if the seBARG proof gets accepted and the ciphertext decrypts to 1.

Clearly, the above protocol satisfies completeness and succinctness as well as it makes the protocol public-coin. Only the first message of the verifier[5] requires private randomness. At first glance, one might think that the above approach should also be sufficient to prove soundness by using similar somewhere extraction techniques that we previously discussed. Unfortunately, this is not the case. There is a subtle technical issue that prevents us from proving its soundness.

seBARGs are not enough. Let us start by stating the high-level strategy for proving soundness. Essentially, we wish to reach a hybrid where we can extract the witness for a single instance from all seBARGs and replace the FHE decryption of the root ciphertext with running the VerifyInstance circuit on the randomness r as in the [Mah18, FHM18, ACGH20] protocols. Our hope is that once we can reach such a hybrid, then we could simply replace the original FHE ciphertext from an encryption of the verifier's secret randomness to an encryption of all zeros. Thus, any successful cheating prover must be breaking the soundness of the underlying non-succinct classic verification protocols [Mah18, FHM18, ACGH20].

The issue that we face is: *how to get to a hybrid where we can replace running the* FHE *decryption on the root ciphertext with running the* VerifyInstance *circuit?* While it might appear to be a minor technical hurdle, it really is a fundamental bottleneck. Basically, in such a hybrid, if we do not rely on FHE decryption at all, then we clearly lose the ability to check whether the "non-extracted" instances have valid proofs or not. That is, since we only have *somewhere extraction*, thus we can only check the validity of the instance that we extract, but not all other instances.

As a result, replacing AoKs with seBARGs is inadequate, due to the lack of full extraction capabilities. The soundness issue outlined above is significant; thus highlights the necessity of using AoKs in the template used by [BKL+22]. One of our main technical contributions is developing a new proof technique to smoothly transition to a hybrid where we no longer need FHE key for the target instance $x_{i^\star}$, but can still check validity of proofs for rest of the instances efficiently.

Bypassing the verification problem via inner layer of FHE bootstrapping. Our approach to navigate around this issue is to consider a separate FHE key for every node of the AND tree. That is, instead of performing entire AND tree computation under a single FHE key, we will put one FHE key at each node. Our intuition is that by carefully modifying the protocol design, we can ensure that we can replace the final verification check from decrypting the root ciphertext under the root's secret key to decrypting FHE ciphertexts at the inner nodes corresponding to a single path (to the cheating leaf node) as per the Merkle-tree-style opening of the FHE AND tree. In this way, we can replace the FHE decryption check for the only the cheating leaf node, while keeping rest of th FHE decryption checks as is. After this, we can replace the encryption of the verifier's randomness to an encryption of all zeros for the cheating instance. Observe that throughout these

[5]Note that the first message can be viewed as a one-time setup or a CRS.

hybrids, we are assured of the validity of all other instances, while we check for the validity of the cheating instance locally.

We summarize our protocol next. The verifier performs a one-time setup, where it samples k FHE public keys, and encrypts the verification randomness for each instance individually. The verifier samples additional FHE public keys corresponding to the inner nodes in the AND tree, and encrypts them to enable desired FHE bootstrapping effect. It sends the set of FHE public keys, encryption of the verifier's randomness, and the FHE bootstrapping ciphertexts to the prover.

1. On receiving all the information, the prover commits to the witnesses and sends an seBARG proof to the verifier that shows that the computation was performed correctly.
2. If the seBARG proof accepts, the verifier samples a random bit c and sends it to the prover.
3. The prover runs the verification circuit under FHE using the encrypted secret key. It then constructs an AND tree of ciphertexts using each FHE public key. The prover sends an seBARG proof that shows that the AND tree was constructed correctly and the ciphertext corresponding to the root node of the AND tree. (Here it crucially relies on the bootstrapping ciphertext to generate the root ciphertext as well as prove correctness of computation of the root ciphertext in a batch**NP**-friendly way.)
4. The verifier accepts if the seBARG proof gets accepted and the root ciphertext decrypts to 1.

The soundness proof for the construction follows the strategy outlined above by carefully organizing the hybrids. Note that all messages (except the first round message) and proofs are succinct, independent of the batch size. The verifier's first round message is quite large, since we need FHE public keys that are twice the size of the number of instances. Thus, this gives us our first result for interactive batch arguments for QMA with long first message, and 3 short messages under the post-quantum hardness of LWE. We provide additional details on both the construction and the soundness proof in Section 5.

We remark that the first round message in the above protocol is reusable, thus a verifier could consider it as a long CRS (i.e., incur a one-time large setup cost for succinct interactive batch arguments for QMA).

Achieving optimal succinctness in 4 rounds via obfuscation. Our final idea is to rely on a post-quantum indistinguishability obfuscator (iO) [BGI+01] to make the verifier's first message succinct. Recall iO allows us to compiler any classical circuit C into an obfuscated circuit $\hat{C}$ such that C and $\hat{C}$ are functionally identical, yet an attacker cannot distinguish $\hat{C}$ from another obfuscated circuit $\hat{C}'$ for any other functionally identical circuit C'.

Note the first verifier message is non-succinct just because the verifier needs to send about $2k$ FHE keys and k classical-quantum commitment keys. Recall the reason the verifier cannot send just one of them is because we are unable to design a reduction in that case. Our insight is that the reduction algorithm in the previous protocol only needs to know about $\log k$ FHE keys at any point during the intermediate hybrids. Thus, the rest of $2k - \log k$ FHE keys could be selected arbitrarily. The reason we cannot simply design $\log k$ keys like that in our previous protocol is because we don't know which $\log k$ out of $2k$ FHE keys will be needed, and moreover, we need to hide our choice of those $\log k$ keys. However, our observation is that we can do possibly do this under the obfuscation hood. That is, we can define a short program that has a PRF key K hardwired, and it uses K to deterministically generate all $2k$ FHE keys (and k commitment keys as well as other auxiliary material). Now by obfuscation security, it will not immediately reveal K.

Thus, by relying on *punctured programming techniques* [SW14], we can simply hardwire the $\log k$ that we need to be securely generated, and leave the rest to be pseudorandomly generated. We provide more details later in Section 6.

We highlight that although we rely on iO in the above overview (as well as later in the main body), our protocol requires iO for only $\log k$-bit inputs (i.e., iO for really short inputs). It turns out for such parameter settings iO can be designed from polynomial hardness of FE [AJ15, BV15, AJS15], which is slightly weaker assumption. It remains an open question on whether we could relax our assumptions further to just post-quantum learning with errors. We remark that recursive composition techniques [BHK17a, KPY19b, WW22] have been useful in the non-interactive setting to reduce the first-message/CRS size, but it remains an intriguing challenge to achieve similar results in the interactive setting. Any progress along this direction might readily give us fully-succinct 4-round batch arguments for QMA.

Additionally, it is also an interesting question to explore whether we could use known correlation intractable hash functions to make these protocols completely non-interactive. We believe that our four-round protocol is very suitable for applying the Fiat-Shamir transform since the verifier's online message is a random string containing the challenge bit for the [Mah18, FHM18, ACGH20] protocol. We leave this an exciting open question for future research.

Generalizing to Monotone Policy batchQMA

The next goal of our work is to go beyond BARGs and design better classical succinct argument for every language in QMA. While the designing 4-round succinct arguments for all of QMA, in the plain model, is beyond the reach of our current work, we have identified a very interesting sub-class of QMA, that we refer to as monotone-batchQMA. Along the theme of our work, we design new succinct arguments for monotone-batchQMA that require only 6-rounds of interaction, under the post-quantum hardness of LWE and in the plain model.

Let us first formally define the class of monotone policy batchQMA. This is a quantum analogue of the class of monotone policy batch**NP** [BBK+23]. Each language $\mathcal{L}_f^{(k)}$ in this class is associated with a predicate f, that can be implemented by a classical monotone circuit C, and another QMA language $\mathcal{L}$ (with a quantum relation $\mathcal{R}$). Formally, it is defined as:

$$\mathcal{L}_f^{(k)} = \left\{ (x_1, \ldots, x_k) : \exists (|\phi_1\rangle, \ldots, |\phi_k\rangle) \text{ s.t } f(b_1, \ldots, b_k) = 1, \text{ where } b_i = \mathcal{R}(x_i, |\phi_i\rangle) \right\}$$

where $|\phi_i\rangle$ is the quantum witness for instance x_i and $\mathcal{R}(x_i, |\phi_i\rangle)$ is used to denote whether $x_i \in \mathcal{L}$. In words, it defines non-deterministic generalization of batchQMA. Here the non-determinism stems from the fact that unlike vanilla batchQMA, where a prover is trying to prove validity of every instance in the batch, a prover corresponding to monotone policy batchQMA is only trying to prove that it knows an input assignment to the monotone circuit such that the circuit accepts, and if i-th bit of input assignment is 1, then the corresponding instance is satisfiable.

A private-coin Monotone BARG for QMA. We initially set our sights on a private coin construction. We recall that the private coin BARG template is to have the prover send a succinct digest of the computation and use an seBARG/AoK to prove the correctness of the computation. An intuitive idea would be to replace each of these seBARG/AoK with a monotone policy BARG by [BBK+23]. Since instance verification occurs in the final round of interaction, monotone SNARGs should enable us to prove that the batch satisfies the monotone circuit C. However, it

remains unclear which monotone circuit should be used for the second and fourth rounds of the protocol, where we essentially prove that the digest was computed correctly.

Upon closer inspection of our private coin BARG construction, we observe that we do not really need seBARGs for a very complicated language. But instead, we only need seBARGs for the the hash membership language. Such seBARGs are commonly referred to as somewhere extractable hash (SEH) functions in the literature. Thus, our first observation is that we could replace the second and fourth rounds seBARG proof with a hash digest computed using an SEH instead. Thus, we only need to use monotone policy BARG for the final round, where the monotone circuit is the same as the circuit for the underlying montone policy batchQMA language.

What circuits to use for monotone SNARGs? Unfortunately, we run into significant challenges while proving the soundness of the above construction. This is in part due to the subtleties that are introduced by the monotone policy BARG. A cheating prover in the previous BARG construction is far more restricted due to the simpler nature of the monotone circuit (essentially an AND tree) compared to a cheating prover for a general monotone circuit C. Indeed, we do not know where to set the index of extraction for the somewhere extractable hash function in order to catch such a cheating prover. This issue was also encountered in the classical setting. To navigate this complication, [BBK+23] use non-uniform reductions, aiming for non-adaptive soundness, and develop a proof system that allowed extraction on an index i from a special set J. The special set J is called the *necessary set* which consists of a subset of indices corresponding to the input wires of the monotone circuit C, that need to be set to one for the circuit to be satisfied. Although a non-uniform reduction could receive the description of such a necessary set J for C, extracting a valid witness from the proof remains an issue.

In [BBK+23], they also introduced a powerful new type of hash function called predicate extractable hash function (PEHash) for bit-fixing predicates to enable such an extraction process. These hash functions allow for the extraction of an index j in the necessary set J along with the extraction for the predicate evaluation, i.e. $f(x)$, on an input x. [BBK+23] also define an extension to PEHash called PEHash with tags (PEHT) where each bit of the input x can have an associated tag t. The PEHT hash function is also extractable on the tag value. Using these two primitives, [BBK+23] were able to appropriately extend somewhere extractability property to predicate extractability for any predicate f with a monotone circuit C.

Armed with this powerful primitive, we can replace the somewhere extractable hash function with a predicate extractable hash function with tags. Let $\mathbf{s}$ denote a string of length k s.t. the jth bit s_j is set to true iff the jth instance belongs to the language $\mathcal{L}$. We use PEHT in the second and fourth rounds of the protocol to hash the value of $\mathbf{s}$ with the commitment string y_j and the opening string z_j as tags for the index j. We retain the monotone SNARG for **NP** as is in the final round. For our reductions, we will make the hash values of $\mathbf{s}$ extractable on the non-uniform advice of the necessary set J and set the bit-fixing predicate string to a string of all zeros. Any cheating prover since must set one of the indices in J to be one to have the circuit C evaluate to one. We can use tag extraction to extract the values and perform the checks locally to catch the cheating prover.

How to handle inconsistent extraction? Although this approach seems to work, there is a fundamental problem with the above construction. *How do we ensure that the witnesses extracted in each round for a cheating prover correspond to the same index from the necessary set J?* Basically, what prevents a cheating prover from using three different values $\mathbf{s}, \mathbf{s}', \mathbf{s}''$ for creating the digest across the three rounds of interaction? The source of this problem is that PEHTs and PEHashs *only* guarantee that the extraction of the index and tag occurs for **some** index j in the necessary set J.

A cheating prover can potentially evade capture by exploiting this property of PEHash extraction, which makes proving the security of the construction difficult.

Our approach way to navigate this issue is to additionally add an accompanying seBARG proof that ensures a cheating prover is committed to use the same string $\mathbf{s}$ in all three rounds. That is, it proves that the value of $\mathbf{s}$ used in each round of the prover computation is consistent. The prover sends such seBARG proofs in rounds four and six, demonstrating that the same value $\mathbf{s}$ was utilized in both rounds two and four, as well as in rounds four and six. Any cheating prover is restricted to produce openings for a consistent value of $\mathbf{s}$ throughout the protocol. Thus, our insight is that by appropriately using soundness property of the seBARGs, we can ensure that a cheating prover can not evade by using different $\mathbf{s}$ values across different rounds.

A final lynchpin to tie together all proofs. While the construction above guards against cheating provers who aim to exploit the nature of index extraction with inconsistent values for $\mathbf{s}$, we still are yet to address the issue of differing extracted indices. We cannot catch any cheating prover if the witnesses are extracted at different locations. In search of more structure within the extraction output, we studied the PEHT construction provided in [BBK+23] more closely. After a careful inspection, we observed that their construction obeys a very special *consistency* property that allows our hybrids to go through. Since [BBK+23] did not require such a consistency property, thus they did not make it explicit. We observe that the index extraction procedure for PEHTs (and by extension PEHash) always extracts the *lexicographically smallest index from the necessary set J, where the input x differs from the bit-fixing predicate string.* This implies that, in our reductions, we can argue that the PEHT extracts the lexicographically smallest index j in the set J, whose value is set to one since the bit-fixing predicate string is set to all zeros. By combining this with the fact that, by carefully using additional seBARG proofs we can ensure consistency across the various rounds of the protocol, we are able to prove soundness of our argument system for monotone policy batchQMA.

We hide a lot of important technical details in the above overview. For instance, we are unable to argue consistency of extracted indices if we use the monotone SNARG construction by [BBK+23] as a black-box. Essentially, we have to open up their construction, argue the desired consistency properties. However, for the purposes of this overview we ignore these details, and defer further discussion to the main body. We refer the reader to Section 7 for more details. The above concludes a brief overview of our 6-round private-coin protocol for monotone-policy batchQMA, without AoK. Unfortunately, unlike for batchQMA, we are unable to go below 6-rounds for monotone-policy batchQMA languages. In part, this is because the Merkle-tree-style FHE computation techniques we relied on for our BARG construction do not generalize as well to monotone-policy batchQMA. We leave this as an interesting open problem.

We remark that we can make our above protocol public-coin by relying on a state-preserving AoK, while keeping at 6-round. While this does improve the current state of the art, we only consider this as a supplementary result, and briefly describe it in Section 8.

Other improvements and related work

Non-interactive protocols in QROM. It is well-known that any public coin interactive protocol can be compiled to a non-interactive protocol by following the Fiat-Shamir transformation under the QROM model [DFMS19, DFM20]. [BKL+22] used the same principle to show that their succinct argument for QMA (and thus BARGs) can be compiled with [DFMS19, DFM20] to get a non-interactive succinct argument of QMA. For completeness, in Section 9, we also show that our

protocols can also be compiled to get non-interactive (monotone policy) BARGs for QMA in the QROM model.

Making protocols publicly verifiable via diO. We also provide a folklore approach to compiler any privately-verifiable (designated-verifier) proof system to a publicly-verifiable proof system using the non-falsifiable assumption of post-quantum Differing Inputs Obfuscation (diO) [Zha23]. The key idea is to have the obfuscated programs of the prover key generation algorithm and the secret key generation step followed by verification algorithm as a part of the CRS and let the prover and verifier run the obfuscated programs. We can achieve non-adaptive soundness by modifying the program to behave differently for the adaptively chosen instance and relying on security of diO for extraction. The details of the construction are given in Section 10.

Related Work. In the classical setting, we have a rich body of results for the various classes of problems that are considered for this work. Starting with [KPY19a] using a non-standard pairing based assumption, we have recent outstanding works by [CJJ21, CJJ22, WW22] in designing BARGs for **NP**. In the pursuit for SNARGs for **NP**, [BBK+23] and [NWW23] build batched arguments for Monotone policy batch**NP** language.

The succinct classical verification of quantum computation, recently formalized by [CCY20, BKL+22], remains an active area of research with recent contributions from [GKNV24] and [MNZ24]. The work by [BKL+22] builds on the Mahadev measurement protocol [Mah18] and achieves an 8 round succinct argument for QMA using state-preserving AoKs and post-quantum secure indistinguishability obfuscation (iO). [GKNV24] extend this previous work to build a succinct and binding commitment scheme to quantum states, facilitating an 8 round public coin succinct argument *without* the need for iO. Orthogonally, [MNZ24] use compiled non-local games to achieve information theoretic question-succinctness and thus also design a classical argument system for QMA. However, [MNZ24] also relies on state-preserving AoKs for their construction, thereby achieving the same round complexity.

3 Preliminaries

Notation We will let PPT denote probabilistically polynomial time and QPT a quantum polynomial time algorithm. We denote the set of all positive integers upto n as $[n] := \{1, 2, \ldots, n\}$. We use the notation $\{0,1\}^n$ to denote binary strings of length n, while we use 1^t to denote a unary string of length t. We denote any polynomial function of an input variable λ as $\mathsf{poly}(\lambda)$. Similarly, we denote all negligible functions as $\mathsf{negl}(\lambda)$. Throughout the paper, we use the notation Primitive.Function to denote the cryptographic primitive and the corresponding function being used. Similarly, we use prim.members with shortened lowercase letters to denote the cryptographic primitive and the members (parameters) being used. The transcript of an interaction between two entities A, B is denoted as $\langle A, B\rangle$ and the output of such a transcript is denoted by $\mathsf{Out}\langle A, B\rangle$.

Quantum Information Let $\mathcal{H}$ be a finite dimensional Hilbert space. A pure state is a unit vector $|\phi\rangle \in \mathcal{H}$. Let $D(\mathcal{H})$ denote the set of all positive semi definite operators on $\mathcal{H}$ with trace 1. A mixed state is an operator $\boldsymbol{\rho} \in D(\mathcal{H})$, and is often called the *density matrix*. We abuse notation slightly and switch between the mixed state representation and the pure state representations. A ℓ-qubit quantum state $\boldsymbol{\rho} \in D(\mathcal{H})$, where $H \simeq (\mathbb{C}^2)^{\otimes \ell}$, we denote the shorthand $M(h, \boldsymbol{\rho})$ as the distribution resulting from measuring each qubit i of $\boldsymbol{\rho}$ in the basis determined by $h \in \{0,1\}^\ell$. By convention, $h_i = 0$ corresponds to a measurement of the ith register in the standard basis $\{|0\rangle, |1\rangle\}$,

while $h_i = 1$ corresponds to a measurement in the Hadamard basis $\{|+\rangle, |-\rangle\}$. We sometimes divide the $\mathcal{H}$ into named registers with pre-set uppercase calligraphy letters, e.g. $\mathcal{H} = \mathcal{A} \otimes \mathcal{B} \otimes \mathcal{C}$ where the function of each register is either explained or implied by the context. We denote the trace operator on a register $\mathcal{A}$ of a quantum state $|\phi\rangle$ as $\mathsf{Tr}_{\mathcal{A}}(|\phi\rangle)$.

3.1 Hash Family with Succinct Local Openings

A Hash Family with Succinct Local Openings (HT) [Mer87] is a tuple of algorithms $(\mathsf{Gen}, \mathsf{Hash}, \mathsf{Open}, \mathsf{Verify})$, where each algorithm is defined as follows.

$\mathsf{Gen}(1^N, 1^\lambda) \to \mathsf{hk}$: A PPT algorithm that uses the security parameter λ and the length of input 1^N to generate the hash key hk.

$\mathsf{Hash}(\mathsf{hk}, \mathbf{y}) \to \hat{y}$: A deterministic poly-time algorithm that computes the hash of input $\mathbf{y} \in \{0,1\}^N$ using hash-key hk to produce a hash-root $\hat{y} \in \{0,1\}^{\mathsf{poly}(\lambda, \log N)}$.

$\mathsf{Open}(\mathsf{hk}, \mathbf{y}, j) \to (b, \mathsf{op})$: A deterministic poly-time algorithm that outputs a bit $b \in \{0,1\}$ and openings $\mathsf{op} \in \{0,1\}^{\mathsf{poly}(\lambda, \log N)}$ to b using the hash-key hk, the input string $\mathbf{y}$, and an index $j \in [N]$.

$\mathsf{Verify}(\mathsf{hk}, \hat{y}, j, b, \mathsf{op}) \to 0/1$: A deterministic poly-time algorithm verifies that the openings (b, op) are valid w.r.t $(\hat{y}, j)$ for some $j \in [N]$.

Definition 3.1. A Hash Family with Succinct Local Openings $\mathsf{HT}(\mathsf{Gen}, \mathsf{Hash}, \mathsf{Open}, \mathsf{Verify})$ satisfies the following properties along with collision resistance with respect to openings and values.

Opening completeness: For all $n \leq 2^\lambda$, for all $\mathbf{y} \in \{0,1\}^n$, for all $j \in [n]$, and for all $\lambda \in \mathbb{N}$, there exists a negligible function $\mathsf{negl}(\cdot)$ s.t

$$\Pr\left[\begin{array}{l} b = y_j \wedge \\ \mathsf{Verify}(\mathsf{hk}, \hat{y}, j, b, \mathsf{op}) = 1 \end{array} : \begin{array}{l} \mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda) \\ \hat{y} \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathbf{y}) \\ (b, \mathsf{op}) \leftarrow \mathsf{Open}(\mathsf{hk}, \mathbf{y}, j) \end{array}\right] = 1 - \mathsf{negl}(\lambda)$$

Computational binding w.r.t openings: For any poly size adversary $\mathcal{A}$, and for all $\lambda \in \mathbb{N}$, there exists a negligible function $\mathsf{negl}(\cdot)$ s.t

$$\Pr\left[\begin{array}{l} \mathsf{Verify}(\mathsf{hk}, \hat{y}, j, 0, \mathsf{op}_0) = 1 \wedge \\ \mathsf{Verify}(\mathsf{hk}, \hat{y}, j, 1, \mathsf{op}_1) = 1 \end{array} : \begin{array}{l} \mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda) \\ (1^n, \hat{y}, j, \mathsf{op}_0, \mathsf{op}_1) \leftarrow \mathcal{A}(\mathsf{hk}) \end{array}\right] = \mathsf{negl}(\lambda)$$

Theorem 3.2 ([Mer87])**.** Assuming the existence of collision resistant hash family, there exists a hash family with local openings (according to Definition 3.1)

3.2 Predicate Extractable Hash functions with Tags for Bit-fixing predicates

A Predicate Extractable Hash Function PEHT [BBK+23] is a hash function that is associated with a bit-fixing predicate class $\mathcal{F}$ and a vector of tags $\vec{t}$ s.t the hash output is **binding** to a specific predicate $f = (J, y) \in \mathcal{F}$ and we can extract $f(x)$, an index $j \in J$ and the tag t associated with location j. The PEHT family is a tuple $(\mathsf{Gen}, \mathsf{Hash}, \mathsf{Open}, \mathsf{Verify}, \mathsf{Extract}, \mathsf{ExtractIndex}, \mathsf{ExtractTag})$ where each algorithm has the description of a monotone circuit C for f as an implicit input. The algorithms are defined as follows:

$\mathsf{Gen}(1^\lambda, N, f = (J, y)) \to (\mathsf{hk}, \mathsf{vk}, \mathsf{td})$: A PPT setup algorithm that takes a security parameter 1^λ, message length N and a bit-fixing predicate $f \in \mathcal{F}$ defined by a set $J \subseteq [N]$ and a string $y \in \{0,1\}^J$. It outputs a hash key hk, a verification key vk, and a trapdoor td.

$\mathsf{Hash}(\mathsf{hk}, x, \vec{t}) \to \mathsf{v}$: A poly-time deterministic algorithm that takes input a hash-key hk, $x \in \{0,1\}^N$, and tags $\vec{t} = (t_1, t_2, \ldots, t_N) \in (\{0,1\}^T)^N$. It outputs a hash-value $\mathsf{v} \in \{0,1\}^{T \cdot \mathsf{poly}(\lambda)}$.

$\mathsf{Open}(\mathsf{hk}, x, \vec{t}, j) \to \rho$: A poly-time deterministic algorithm that takes as input a hash key hk, $x \in \{0,1\}^N$, tags $\vec{t} = (t_1, t_2, \ldots, t_N)$, and an index $j \in [N]$ to output an opening $\rho \in \{0,1\}^{T \cdot \mathsf{poly}(\lambda)}$.

$\mathsf{Verify}(\mathsf{vk}, \mathsf{v}, j, b, t, \rho) \to 0/1$: A poly-time deterministic algorithm that checks if the opening ρ is valid w.r.t. (v, j, b, t) for some $j \in [N]$

$\mathsf{Extract}(\mathsf{td}, \mathsf{v}) \to u$: A deterministic extraction algorithm that uses the trapdoor td and the hash value $\mathsf{v} \in \{0,1\}^{T \cdot \mathsf{poly}(\lambda)}$ to output a bit u.

$\mathsf{ExtractIndex}(\mathsf{td}, \mathsf{v}) \to j$: A deterministic extraction algorithm that uses the trapdoor td and the hash value $\mathsf{v} \in \{0,1\}^{T \cdot \mathsf{poly}(\lambda)}$ and outputs an index $j \in [N]$

$\mathsf{ExtractTag}(\mathsf{td}, \mathsf{v}) \to t$: A deterministic extraction algorithm that uses the trapdoor td and the hash value $\mathsf{v} \in \{0,1\}^{T \cdot \mathsf{poly}(\lambda)}$ and outputs a tag $t \in \{0,1\}^T$

Definition 3.3 (PEHT)**.** A Predicate Extractable Hash Family with Tags w.r.t. a bit-fixing predicate class $\mathcal{F}$, is the tuple $\mathsf{PEHT}(\mathsf{Gen}, \mathsf{Hash}, \mathsf{Open}, \mathsf{Verify}, \mathsf{Extract}, \mathsf{ExtractIndex}, \mathsf{ExtractTag})$, satisfies the following that properties

Opening Completeness: For any $\lambda \in \mathbb{N}$, any $N \leq 2^\lambda$, any predicate $f \in \mathcal{F}$, any index $j \in [N]$, any set of tags $\vec{t} = (t_1, t_2, \ldots, t_N) \in \{0,1\}^{T \cdot N}$, and any $x \in \{0,1\}^N$, we must have

$$\Pr\left[\begin{array}{l|ll} \mathsf{Verify}(\mathsf{vk}, \mathsf{v}, j, x_j, t_j, \rho) = 1 & : & (\mathsf{hk}, \mathsf{vk}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda, N, f) \\ & & \mathsf{v} = \mathsf{Hash}(\mathsf{hk}, x, \vec{t}) \\ & & \rho = \mathsf{Open}(\mathsf{hk}, x, \vec{t}, j) \end{array}\right] = 1$$

Succinctness: In the completeness experiment above, the size of the verification key vk and the hash value v is $\mathsf{poly}(\lambda)$. The size of the hash key hk is at most $|f| \cdot \mathsf{poly}(\lambda, \log N)$

Computational binding w.r.t openings: For any poly size adversary $\mathcal{A}$, any polynomial $N = N(\lambda)$, and any predicate $f \in \mathcal{F}$, there exists a negligible function $\mathsf{negl}(\cdot)$ s.t for all $\lambda \in \mathbb{N}$

$$\Pr\left[\begin{array}{l|ll} \mathsf{Verify}(\mathsf{vk}, \mathsf{v}, j, 0, t, \rho_0) = 1 \wedge & : & (\mathsf{hk}, \mathsf{vk}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda, N, J, y) \\ \mathsf{Verify}(\mathsf{vk}, \mathsf{v}, j, 1, t, \rho_1) = 1 & & (\mathsf{v}, j, \rho_0, t, \rho_1) \leftarrow \mathcal{A}(\mathsf{hk}, \mathsf{vk}) \end{array}\right] = \mathsf{negl}(\lambda)$$

Prediate Hiding: For any poly-sized adversary $\mathcal{A}$, any polynomial $N = N(\lambda)$, and any two predicates $f_0, f_1 \in \mathcal{F}$ s.t $|f_0| = |f_1|$, there exists a negligible function $\mathsf{negl}(\cdot)$ s.t for every $\lambda \in \mathbb{N}$, we have

$$\Pr\left[\begin{array}{l|ll} \mathcal{A}(\mathsf{hk}, \mathsf{vk}) = b & : & b \leftarrow \{0,1\} \\ & & (\mathsf{hk}, \mathsf{vk}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda, N, f_b) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

Extraction Correctness: For any $\lambda \in \mathbb{N}$, any $N \leq 2^\lambda$, any predicate $f \in \mathcal{F}$, any set of tags $\vec{t} = (t_1, t_2, \ldots, t_N) \in \{0,1\}^{T \cdot N}$, any $x \in \{0,1\}^N$, there exists a negligible function $\mathsf{negl}(\cdot)$, s.t for every $\lambda \in \mathbb{N}$, we must have

$$\Pr\left[f(x) \neq \mathsf{Extract}(\mathsf{td}, \mathsf{v}) \ : \ \begin{array}{l} (\mathsf{hk}, \mathsf{vk}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda, N, f) \\ \mathsf{v} = \mathsf{Hash}(\mathsf{td}, x, \vec{t}) \end{array} \right] \leq \mathsf{negl}(\lambda)$$

Index Extraction Correctness: For any $\lambda \in \mathbb{N}$, any $N \leq 2^\lambda$, any predicate $f = (J, y) \in \mathcal{F}$ s.t $J \neq \emptyset$, and any hash value $\mathsf{v} \in \{0,1\}^{T \cdot \mathsf{poly}(\lambda)}$, we must have

$$\Pr\left[\mathsf{ExtractIndex}(\mathsf{td}, \mathsf{v}) \in J \ : \ (\mathsf{hk}, \mathsf{vk}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda, N, J, y) \right] = 1$$

Consistency of Extraction: For any poly-sized adversary $\mathcal{A}$, it holds for any polynomial $N = N(\lambda)$ and any bit-fixing predicate described by a set $J \subset [N]$ and a string $y \in \{0,1\}^J$, there exists a negligible function $\mathsf{negl}(\cdot)$, s.t for any $\lambda \in \mathbb{N}$,

$$\Pr\left[\begin{array}{l} j \in J \ \wedge \ \mathsf{Extract}(\mathsf{td}, \mathsf{v}) = 1 \ \wedge \\ \mathsf{Verify}(\mathsf{vk}, \mathsf{v}, j, 1 - y_j, t, \rho) = 1 \end{array} \ : \ \begin{array}{l} (\mathsf{hk}, \mathsf{vk}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda, N, J, y) \\ (\mathsf{v}, j, t, \rho) \leftarrow \mathcal{A}(\mathsf{hk}, \mathsf{vk}) \end{array} \right] \leq \mathsf{negl}(\lambda)$$

and

$$\Pr\left[\begin{array}{l} \mathsf{Extract}(\mathsf{td}, \mathsf{v}) = 0 \ \wedge \\ \mathsf{ExtractIndex}(\mathsf{td}, \mathsf{v}) = j \ \wedge \\ \mathsf{Verify}(\mathsf{vk}, \mathsf{v}, j, y_j, t, \rho) = 1 \end{array} \ : \ \begin{array}{l} (\mathsf{hk}, \mathsf{vk}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda, N, J, y) \\ (\mathsf{v}, j, t, \rho) \leftarrow \mathcal{A}(\mathsf{hk}, \mathsf{vk}) \end{array} \right] \leq \mathsf{negl}(\lambda)$$

Consistency of Tag Extraction: For any poly-sized adversary $\mathcal{A}$, it holds for any polynomial $N = N(\lambda)$ and any bit-fixing predicate described by a set $J \subseteq [N]$ and a string $y \in \{0,1\}^J$, there exists a negligible function $\mathsf{negl}(\cdot)$ s.t for any $\lambda \in \mathbb{N}$

$$\Pr\left[\begin{array}{l} \mathsf{Extract}(\mathsf{td}, \mathsf{v}) = 0 \ \wedge \\ \mathsf{ExtractIndex}(\mathsf{td}, \mathsf{v}) = j \ \wedge \\ \mathsf{ExtractTag}(\mathsf{td}, \mathsf{v}) \neq t \ \wedge \\ \mathsf{Verify}(\mathsf{vk}, \mathsf{v}, j, 1 - y_j, t, \rho) = 1 \end{array} \ : \ \begin{array}{l} (\mathsf{hk}, \mathsf{vk}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda, N, J, y) \\ (\mathsf{v}, j, t, \rho) \leftarrow \mathcal{A}(\mathsf{hk}, \mathsf{vk}) \end{array} \right] \leq \mathsf{negl}(\lambda)$$

Remark 3.4. [BBK+23] also define the notion of the Predicate Extractable Hash family for Bit Fixing predicates (without the tags) as follows $\mathsf{PEHash}(\mathsf{Gen}, \mathsf{Hash}, \mathsf{Open}, \mathsf{Verify}, \mathsf{Extract}, \mathsf{ExtractIndex})$. The syntax for the PEHash family remains the same as that of PEHT, but we omit the need for tags. Alternatively, PEHash can be considered as a PEHT family with each tag as $\perp$. The PEHash family satifies all the properties defined in Definition 3.3 except the Consistency of Tag Extraction.

Remark 3.5. PEHTs [BBK+23] also satisfy a **Lexicographically Smallest Index Extraction** property by correctness of FHE. We define and prove this property in Section 7.2.

3.3 Somewhere Extractable BARG (seBARG) for NP

A Somewhere Extractable Batch Argument or seBARG [CJJ22] for an **NP** language $\mathcal{L}$ (corresponding relation $\mathcal{R}$) is a tuple $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Extract})$, where each algorithm is defined as follows.

$\mathsf{Setup}(1^\lambda, 1^k, 1^n, i^*) \to (\mathsf{crs}, \mathsf{td})$: A PPT algorithm that takes security parameter 1^λ, batch size k, input length n, and index $i^* \in [k]$. It outputs a crs and trapdoor td.

$\mathsf{Prove}(\mathsf{crs}, \{x_j\}_{j\in[k]}, \{w_j\}_{j\in[k]}) \to \pi$: A PPT algorithm takes in the crs, the batch of statements $x_1, \ldots, x_k$ with corresponding witnesses $w_1, \ldots, w_k$ and produces proof π.

$\mathsf{Verify}(\mathsf{crs}, \{x_j\}_{j\in[k]}, \pi) \to 0/1$: A PPT algorithm that verifies the proof π generated by Prove and outputs 1 if the proof is valid (0 if not).

$\mathsf{Extract}(\mathsf{td}, \{x_j\}_{j\in[k]}, \pi) \to w^*$: A PPT algorithm that takes in a trapdoor td, an index $i^* \in [k]$, proof π, and extracts the corresponding witness w^* at the index specified by i^* during crs generation.

Definition 3.6 (seBARG)**.** A somewhere-extractable batch argument scheme (seBARG) for $\mathcal{L}$ is required to satisfy the following properties:

Efficiency: The length of the crs and proof π is at most $\mathsf{poly}(\lambda, \log k, n, m)$, where m is the length of a witness.

Completeness: For any $k = k(\lambda)$ instances $x_1, \ldots, x_k$ and witnesses $w_1, \ldots, w_k$ of length m s.t $(x_i, w_i)_{i\in[k]} \in \mathcal{R}$, any index $i^* \in [k]$, any $\lambda \in \mathbb{N}$, and any polynomial $n = n(\lambda)$, we must have

$$\Pr\left[\begin{array}{c|l} \mathsf{Verify}(\mathsf{crs}, \{x_j\}_{j\in[k]}, \pi) = 1 & (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda, 1^k, 1^n, i^*) \\ & \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, \{x_j\}_{j\in[k]}, \{w_j\}_{j\in[k]}) \end{array}\right] = 1$$

Index Hiding: For any polynomial time adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$, s.t $\forall\lambda \in \mathbb{N}$ we have,

$$\Pr\left[\begin{array}{c|l} b' = b & i_0, i_1 \in [k],\ b \leftarrow \{0,1\}, \\ & (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda, 1^k, 1^n, i_b) \\ & b' \leftarrow \mathcal{A}(\mathsf{crs}) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

Somewhere Argument of Knowledge: For any polynomial time adversary $\mathcal{A}$ and for any index $i^* \in [k]$, the witness extracted at i^* using the trapdoor td is a valid witness, i.e there exists a negligble function $\mathsf{negl}(\cdot)$ s.t

$$\Pr\left[\begin{array}{c|l} \mathsf{Verify}(\mathsf{crs}, \{x_j\}_{j\in[k]}, \pi) = 1 \wedge & i^* \leftarrow \mathcal{A} \\ (x_{i^*}, w^*) \notin \mathcal{R} & (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda, 1^k, 1^n, i^*) \\ & (x_1, x_2, \ldots, x_k, \pi) \leftarrow \mathcal{A}(\mathsf{crs}) \\ & w^* \leftarrow \mathsf{Extract}(\mathsf{td}, i^*, \pi) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

3.4 Fully Homomorphic Encryption

A Fully Homomorphic Encryption (FHE) scheme allows for computation to be performed on encrypted data. Let $\mathcal{C} = \{C_\lambda\}_{\lambda\in\mathbb{N}}$ with n input wires be an ensemble of circuits. A FHE-scheme for $\mathcal{C}$ is a tuple $(\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Enc}, \mathsf{Dec})$, where each algorithm is defined as follows

$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$: A PPT algorithm that takes a security parameter λ and outputs a public/secret key pair.

$\mathsf{Enc}(\mathsf{pk}, x) \to \mathsf{ct}$: A PPT algorithm that takes message $x \in \{0,1\}^n$ and produces ciphertext ct.

$\mathsf{Eval}(\mathsf{pk}, C, \mathsf{ct}) \to \hat{\mathsf{ct}}$: A PPT algorithm that takes ciphertext ct and outputs ciphertext $\hat{\mathsf{ct}}$

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to x$: A deterministic algorithm that takes the secret key sk and a ciphertext ct to output a corresponding message x.

Definition 3.7 (FHE Properties)**.** Let $\mathcal{C} = \{C_\lambda\}_{\lambda\in\mathbb{N}}$ be an ensemble of circuits. A FHE scheme $(\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Enc}, \mathsf{Dec})$ for $\mathcal{C}$ satisfies the following properties

Correctness: For any circuit $C \in \mathcal{C}$, for every $x \in \{0,1\}^n$, for any $\lambda \in \mathbb{N}$ we must have

$$\Pr\left[\begin{array}{c|l} C(x) = \alpha & \begin{array}{l}(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)\\ c \leftarrow \mathsf{Enc}(\mathsf{pk}, x)\\ c' \leftarrow \mathsf{Eval}(\mathsf{pk}, C, c)\\ \alpha \leftarrow \mathsf{Dec}(\mathsf{sk}, c')\end{array}\end{array}\right] = 1$$

Compactness: The size of $\mathsf{Eval}(\mathsf{pk}, C, \mathsf{Enc}(\mathsf{pk}, x))$ is fixed $\mathsf{poly}(\lambda)$ and is independent of $|C|, |x|$ for all circuits $C \in \mathcal{C}$.

Semantic Security: For every poly-size adversary $\mathcal{A}$ and for every $\lambda \in \mathbb{N}$, there exists a negligible function $\mathsf{negl}(\cdot)$ s.t

$$\Pr\left[\begin{array}{c|l} b' \leftarrow \mathcal{A}(\mathsf{ct}) \wedge b = b' & \begin{array}{l}(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)\\ x_0, x_1 \leftarrow \mathcal{A}(\mathsf{pk})\\ b \in_R \{0,1\}\\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, x_b)\end{array}\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

Theorem 3.8 (Post-quantum security of FHE [BKL+22])**.** Under circular-secure variants of the learning with errors assumption, there exists a fully homomorphic encryption scheme for all polynomial-time computable functions. If the circular LWE variant is post-quantum, then so is the FHE scheme.

Under the standard LWE assumption, there exists a FHE scheme for all polynomial-size circuits of depth $d(\lambda)$, where the scheme has compactness $\mathsf{poly}(\lambda, d)$.

3.5 Puncturable Pseudo-random Functions (PPRF)

Definition 3.9 (Puncturable PRFs [BW13, BGI14, KPTZ13, ?])**.** A puncturable PRF family is a family of functions

$$\mathcal{PF} = \left\{F_{\lambda,s} : \{0,1\}^{\nu(\lambda)} \to \{0,1\}^{\mu(\lambda)}\right\}_{\lambda\in\mathbb{N},\ s\in\{0,1\}^{\ell(\lambda)}}$$

with the associated tuple of algorithms $\mathcal{PF}(\mathsf{Eval}, \mathsf{Puncture}, \mathsf{PuncEval})$ satisfying

1. For all $x \in \{0,1\}^{\nu(\lambda)}$ and all $s \in \{0,1\}^{\ell(\lambda)}$, $\mathcal{PF}.\mathsf{Eval}(s,x) = F_{\lambda,s}(x)$.
2. For all distinct $x, x' \in \{0,1\}^{\nu(\lambda)}$ and all $s \in \{0,1\}^{\ell(\lambda)}$,

$$\mathcal{PF}.\mathsf{PuncEval}(\mathcal{PF}.\mathsf{Puncture}(s,x), x') = \mathcal{PF}.\mathsf{Eval}(s,x')$$

For ease of notation, we will use $F_s(x)$ to denote $\mathcal{PF}.\mathsf{Eval}(s,x)$, and we write $s\{x\}$ to denote $\mathcal{PF}.\mathsf{Puncture}(s,x)$. We define the security of PPRFs as follows

Definition 3.10 (Security of PPRFs)**.** $\mathcal{PF}$ is said to be (s,δ) secure if for every $\{x^{(\lambda)} \in \{0,1\}^{\nu(\lambda)}\}_{\lambda\in\mathbb{N}}$, the following two distribution ensembles (indexed by λ) are $\delta(\lambda)$ indistinguishable to circuits of size $s(\lambda)$:

$$\left(S\{x^{(\lambda)}\}, F_S(x^{(\lambda)})\right) \qquad \text{and} \qquad \left(S\{x^{(\lambda)}\}, U\right)$$

where $S \leftarrow \{0,1\}^{\ell(\lambda)}$ and $U \leftarrow \{0,1\}^{\mu(\lambda)}$.

Theorem 3.11 ([BW13, BGI14, KPTZ13, ?, GGM86])**.** If { polynomially secure, subexponentially secure } one-way functions exist, then for all $\mu : \mathbb{N} \to \mathbb{N}$ (with $1^{\nu(\lambda)}$ polynomial computable from 1^{ν}), and all $\delta : \mathbb{N} \to [0,1]$ with $\delta(\nu) \leq 2^{-\mathsf{poly}(\nu)}$, there are polynomials $\ell(\lambda), \nu(\lambda)$ and a { polynomially, $\left(\frac{1}{\delta(\nu(\lambda))}, \delta(\nu(\lambda))\right)$-secure } puncturable PRF family

$$\mathcal{PF}_\mu = \left\{F_{\lambda,s} : \{0,1\}^{\nu(\lambda)} \to \{0,1\}^{\mu(\nu(\lambda))}\right\}_{\lambda\in\mathbb{N},\ s\in\{0,1\}^{\ell(\lambda)}}$$

3.6 Indistinguishability Obfuscation (iO)

Definition 3.12 ($\mathsf{i}\mathcal{O}$)**.** An indistinguishability obfuscator is an algorithm $\mathsf{i}\mathcal{O}$ that takes input a circuit C and satisfies the following properties

1. **Functional Equivalence**: for any polynomial size circuit $C : \{0,1\}^n \to \{0,1\}^m$ and any input $x \in \{0,1\})^n$, we have
$$\mathsf{i}\mathcal{O}(C)(x) = C(x)$$
2. **Security**: For any pair of functionally equivalent circuits (C_0, C_1), we have
$$\mathsf{i}\mathcal{O}(C_0) \approx_c \mathsf{i}\mathcal{O}(C_1)$$

Remark 3.13. We use the candidates for $\mathsf{i}\mathcal{O}$ that are plausibly post-quantum secure [BGMZ18, CVW18, BDGM20, WW21, GP21, DQV+21]. We also use the perfect completeness variant [BV17] to simplify our analysis.

3.7 Differing Inputs Obfuscation (diO)

There are various notions of distributional Differing Inputs Obfuscation $\mathsf{di}\mathcal{O}$ for classical adversaries in the literature [BCP14, ABG+13]. Previous attempts at defining $\mathsf{di}\mathcal{O}$ for the quantum setting were done in [GZ20, CLLZ21] on the closely related topic of *extractable witness encryption*. We will be considering the definition presented in [Zha23] as it compiles well with our constructions. We define $\mathsf{di}\mathcal{O}$ as follows

Definition 3.14 (Post-quantum Differing Inputs Obfuscation $\mathsf{di}\mathcal{O}$)**.** A post quantum *Differing Inputs Obfuscator* is a PPT algorithm $\mathsf{di}\mathcal{O}$ s.t

1. $\mathsf{di}\mathcal{O}(1^\lambda, C)$ is equivalent to C with overwhelming probability over the randomness of $\mathsf{di}\mathcal{O}$.
2. For every QPT adversary $\mathcal{A}$ and sampler $\mathcal{S}$, there exists a QPT extractor $\mathcal{E}$ s.t for every inverse polynomial function $\epsilon = \epsilon(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ s.t

$$\Pr\left[\mathsf{Fail}_\epsilon^{\mathsf{di}\mathcal{O}}\right] < \mathsf{negl}(\lambda), \qquad \Pr\left[\mathsf{Live}_\epsilon^{\mathsf{di}\mathcal{O}} \wedge \neg\, \mathsf{Success}_\epsilon^{\mathsf{di}\mathcal{O}}\right] < \mathsf{negl}(\lambda)$$

   where $\mathsf{Fail}_\epsilon^{\mathsf{di}\mathcal{O}}, \mathsf{Live}_\epsilon^{\mathsf{di}\mathcal{O}}, \mathsf{Success}_\epsilon^{\mathsf{di}\mathcal{O}}$ are defined w.r.t a security game as defined below

Definition 3.15 (Security Game for $\mathsf{di}\mathcal{O}$)**.** Let $m(\lambda), n(\lambda), s(\lambda)$ be polynomials. Consider any QPT sampler $\mathcal{S}(1^\lambda)$ which samples a pair of classical circuits C_0, C_1 of size $s(\lambda)$, input length $n(\lambda)$, and output length $m(\lambda)$, and an auxiliary quantum side information aux. Consider any quantum distinguishing adversary $\mathcal{A}$. Consider any distinguishing operation on aux as follows:

1. Choose a random bit $b \leftarrow \{0,1\}$
2. Run $\hat{C} \leftarrow \mathsf{di}\mathcal{O}(1^\lambda, C_b)$ to get the obfuscation of a random choice between C_0 and C_1.
3. Run $b' \leftarrow \mathcal{A}(\hat{C}, C_0, C_1, \mathsf{aux})$
4. Output 1 iff $b' = b$, otherwise output 0

Let $\mathcal{M}^{\mathsf{di}\mathcal{O}} = \{M_0, M_1\}$ be the POVM given by this operation. [Zha23] uses a projective implementation of $\mathcal{M}^{\mathsf{di}\mathcal{O}} = \{M'_p\}_p$ where each M'_p corresponds to the probability distribution on $\{0,1\}$ that is 1 with probability p. Consider an extractor $\mathcal{E}$ given inputs $(C_0, C_1, \mathsf{aux}, 1^{1/\epsilon})$ for any inverse polynomial function $\epsilon(\lambda)$. The security game for $\mathsf{di}\mathcal{O}$ characterized by $\mathcal{S}, \mathcal{A}, \mathcal{E}$ as well as the function $\epsilon(\lambda)$ is defined as follows:

1. Run $(C_0, C_1, \mathsf{aux}) \leftarrow \mathcal{S}(1^\lambda)$. Let $\mathcal{H}$ be the register containing aux.
2. Apply the projective implementation of $\mathcal{M}^{\mathsf{di}\mathcal{O}}$ to $\mathcal{H}$, resulting in probability p.
3. If $p > \frac{1}{2} + \epsilon(\lambda)$, the event $\mathsf{Live}_\epsilon^{\mathsf{di}}\mathcal{O}$ has occured.
4. Run $\mathcal{E}$ on $\mathcal{H}$, obtaining an input x or the symbol $\perp$.
5. If $x \neq \perp$ but $C_0(x) = C_1(x)$, the event $\mathsf{Fail}_\epsilon^{\mathsf{di}}\mathcal{O}$ has occured.
6. If $x \neq \perp$ and $C_0(x) \neq C_1(x)$, the event $\mathsf{Success}_\epsilon^{\mathsf{di}}\mathcal{O}$ has occured.

3.8 State Preserving Arguments of Knowledge

To establish the soundness of our public-coin Monotone policy BARG and similar constructions from [BKL+22, GKNV24, MNZ24], a full extraction of the witness is required. Traditionally, extraction involves rewinding a successful attacker to retrieve a claimed witness, but with quantum states, rewinding risks irreversibly altering the prover state, resulting in flawed extraction. However, a recent development in [LMS22] introduces a state-preserving argument of knowledge, enabling complete extraction without compromising the quantum state. We define it as follows:

Definition 3.16 (ϵ-State Preserving Arguments of Knowledge AoK)**.** A publicly verifiable argument system Π for an NP language $\mathcal{L}$ (with the corresponding relation $\mathcal{R}_{\mathcal{L}}$) is an ϵ-state preserving argument-of-knowledge if it satisfies the following properties

> **Succinctness**: When invoked on a security parameter $\lambda \in \mathbb{N}$ and an instance n and a relation that is decidable in time T, the communication complexity of the protocol is $\mathsf{poly}(\lambda, \log T)$. The computational complexity of the verifier is $\mathsf{poly}(\lambda, \log T) + \tilde{\mathcal{O}}(n)$.
>
> **ϵ-State-Preserving Extraction**: There exists an extractor $E^{(\cdot)}(x, \epsilon)$ with the following properties.
>
> - **Efficiency**: $E^{(\cdot)}(x, \epsilon)$ runs in time $\mathsf{poly}(\lambda, n, 1/\epsilon)$ as a quantum oracle algorithm (with the ability to apply controlled U-gates given an oracle $U(\cdot)$). It outputs a classical transcript $\tilde{\tau}$ and a classical string w
> - **State Preserving**: Let $|\psi\rangle \in \mathcal{A} \otimes \mathcal{I}$ be any $\mathsf{poly}(\lambda)$-qubit pure state and let $\boldsymbol{\rho} = \mathsf{Tr}_{\mathcal{A}}(|\psi\rangle) \in D(\mathcal{I})$. Consider the following two games:
>   - *Game 0 (Real)*: Generate a transcript τ by running $\mathsf{P}^{\star}(\boldsymbol{\rho}_{\mathcal{I}}, x)$ with the honest verifier V. Output τ along with the residual state on $\mathcal{A} \otimes \mathcal{I}$.
>   - *Game 1 (Simulated)*: Generate a transcript-witness pair $(\tilde{\tau}, w) \leftarrow E^{\mathsf{P}^{\star}(\boldsymbol{\rho}_{\mathcal{I}}, x)}$. Output $\tilde{\tau}$ and the residual state on $\mathcal{A} \otimes \mathcal{I}$
>
>   The output distributions of *Game 0* and *Game 1* are computationally ϵ-indistinguishable to any quantum distinguisher
> - **Extraction Correctness**: For any $\mathsf{P}^{\star}$ as above, the probability that $\tilde{\tau}$ is an accepting transcript but $\mathcal{R}_{\mathcal{L}}(x, w) = 0$ is at-most $\epsilon + \mathsf{negl}(\lambda)$

Theorem 3.17 ([LMS22])**.** Assuming the post-quantum $\mathsf{poly}(\lambda, 1/\epsilon)$ hardness of Learning with Errors LWE, there exists a (4 message, public coin) ϵ-state preserving succinct argument of knowledge.

3.9 Classical Verification for QMA

In this subsection, we define the class QMA and the building blocks required to construct a classical verification protocol. This includes the Mahadev measurement protocol [Mah18] and the non-interactive post hoc verification protocol [FHM18] (referred to as the FHM) protocol.

3.9.1 The Complexity Class QMA

The Quantum Merlin-Arthur (QMA) complexity class can be roughly construed as the quantum analogue of **NP**. QMA consists of decision problems $\mathcal{L} = (\mathcal{L}_{\text{yes}}, \mathcal{L}_{\text{no}})$ between a computationally

powerful quantum prover: Merlin and a less computationally powerful quantum verifier: Arthur. Let the poly-sized quantum circuits corresponding to Merlin and Arthur be P and V, respectively. For any statement $x \in \{0,1\}^n$ in the language $\mathcal{L}$ and quantum proof $|\phi\rangle$ on $p(n)$ qubits, we must have:

Completeness: For all $x \in \mathcal{L}_{\text{yes}}$, there exists a quantum witness $|\psi\rangle$ on at most $p(n)$ qubits

$$\Pr\left[\mathsf{V}(|\phi\rangle, x) = 1 : |\phi\rangle \leftarrow \mathsf{P}(|\psi\rangle, x)\right] \geq 2/3$$

Soundness: For all $x \in \mathcal{L}_{\text{no}}$ and all quantum states $|\psi\rangle$ on at most $p(n)$ qubits,

$$\Pr\left[\mathsf{V}(|\phi\rangle, x) = 1 : |\phi\rangle \leftarrow \mathsf{P}(|\psi\rangle, x)\right] \leq 1/3$$

3.9.2 Interactive Arguments for QMA

Definition 3.18 ([BKL+22])**.** An interactive argument (P, V) for a language $\mathcal{L} = (\mathcal{L}_{yes}, \mathcal{L}_{no}) \in \mathsf{QMA}$ with relation $\mathcal{R}(x)$ is a (classical) 2-party interactive protocol between a QPT prover P and a PPT verifier V, with the following completeness and soundness guarantees:

Completeness: For all $x \in \mathcal{L}_{\text{yes}}$, for all $|\phi\rangle \in \mathcal{R}(x)$, for all $\lambda \in \mathbb{N}$ and polynomials $k = k(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that:

$$\Pr\left[\mathsf{Out}\left\langle \mathsf{P}(|\phi\rangle^{\otimes k(\lambda)}, x), \mathsf{V}(x)\right\rangle = 1\right] \geq 1 - \mathsf{negl}(\lambda)$$

where $\langle \mathsf{P}, \mathsf{V}\rangle$ denotes the process of interaction between P and V

Soundness: For all $x \in \mathcal{L}_{\text{no}}$ and all non-uniform QPT provers P^*, we must have $\forall \lambda \in \mathbb{N}$

$$\Pr\left[\mathsf{Out}\left\langle \mathsf{P}^*(x), \mathsf{V}(x)\right\rangle = 1\right] \leq \mathsf{negl}(\lambda)$$

3.9.3 Interactive Batch Arguments for QMA

We define Batch arguments for QMA and expect it to satisfy the following properties

Definition 3.19 (BARG Completeness)**.** An interactive argument (P, V) for a batch language $\mathcal{L} = \mathcal{L}_1 \times \cdots \times \mathcal{L}_k \in \mathsf{QMA}$ with relation $\mathcal{R}(x)$ satisfies completeness if for any honest prover P, for all $\lambda \in \mathbb{N}$ and all polynomials $k = k(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that:

$$\Pr\left[\mathsf{Out}\left\langle \mathsf{P}(x), \mathsf{V}(x)\right\rangle = 1 \ : \ \forall i \in [k] \ \ x_i \in \mathcal{L}_{\text{yes}}\right] \geq 1 - \mathsf{negl}(\lambda)$$

Definition 3.20 (BARG Semi-adaptive Soundness)**.** An interactive argument (P, V) for a batch language $\mathcal{L} = \mathcal{L}_1 \times \cdots \times \mathcal{L}_k \in \mathsf{QMA}$ with relation $\mathcal{R}(x)$ is semi-adaptively-sound if for any quantum adversary $\mathcal{A}$, for all $\lambda \in \mathbb{N}$ and all polynomials $k = k(\lambda)$, there exists an index $i^* \in [k]$ and there exists a negligible function $\mathsf{negl}(\cdot)$ such that:

$$\Pr\left[\mathsf{Out}\left\langle \mathsf{P}^*(x), \mathsf{V}(x)\right\rangle = 1 \ : \ x_{i^*} \in \mathcal{L}_{\text{no}}\right] \leq \mathsf{negl}(\lambda)$$

3.9.4 Commit-and-Measure Protocols

The notion of *commit and measure procotols* was first formally introduced in [Mah18]. However, we use the abstraction presented in [BKL+22] for our constructions. The protocol allows a classical verifier to obtain measurement results of an $N-$qubit quantum state $\boldsymbol{\sigma}$ held by an untrusted quantum-capable prover.

Definition 3.21 (Commit-and-Measure Protocol Syntax [BKL+22])**.** An $N-$qubit commit-and-measure protocol between a quantum polynomial-time prover $\mathsf{P} = (\mathsf{Com}, \mathsf{Open})$ and a classical probabilistic polynomial-time verifier $\mathsf{V} = (\mathsf{Gen}, \mathsf{Test}, \mathsf{Out})$ has the following syntax.

1. The verifier samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, C)$, where $C : [N] = \{0,1\}^{\log N} \to \{0,1\}$ represents a basis vector $h \in \{0,1\}^N$, obtaining public parameters pk and secret parameters sk. It sends the public parameters pk to the verifier.

2. The prover computes $(y, \boldsymbol{\rho}) \leftarrow \mathsf{Com}(\mathsf{pk}, \boldsymbol{\sigma})$, obtaining a classical "commitment" string y and a private quantum state $\boldsymbol{\rho}$. It sends y to the verifier.

3. The verifier samples a random challenge bit $c \leftarrow \{0,1\}$ and sends c to the prover; $c = 0$ corresponds to a "test round" and $c = 1$ corresponds to a "measurement round".

4. The prover computes $z \in \mathsf{Open}(\boldsymbol{\rho}, c)$, obtaining a classical string z that it sends to the verifier.

5. If $c = 0$, the verifier computes $\{\mathsf{acc}, \mathsf{rej}\} \leftarrow \mathsf{Test}(\mathsf{pk}, (y, z))$.
   If $c = 1$, the verifier computes $m \leftarrow \mathsf{Open}(\mathsf{sk}, (y, z))$ to obtain a classical string $m \in \{0,1\}^N$ of measurement outcomes.

Definition 3.22 (Commit-and-Measure Protocol Completeness [BKL+22])**.** A commit-and-measure protocol is required to satisfy the two completeness properties.

1. (Test Round Completeness) For all $C : [N] \to \{0,1\}$ and $N-$qubit states $\boldsymbol{\sigma}$:

$$\Pr\left[\mathsf{Test}(\mathsf{pk}, (y,z)) = \mathsf{acc} \;:\; \begin{array}{c}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, C)\\ (y, \boldsymbol{\rho}) \leftarrow \mathsf{Com}(\mathsf{pk}, \boldsymbol{\sigma})\\ z \leftarrow \mathsf{Open}(\boldsymbol{\rho}, 0)\end{array}\right] = 1 - \mathsf{negl}(\lambda)$$

2. (Measurement Round Completeness) For all $C : [N] \to \{0,1\}$ and $N-$qubit states $\boldsymbol{\sigma}$:

$$\Pr\left[\mathsf{Out}(\mathsf{sk}, (y,z)) = m \;:\; \begin{array}{c}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, C)\\ (y, \boldsymbol{\rho}) \leftarrow \mathsf{Com}(\mathsf{pk}, \boldsymbol{\sigma})\\ z \leftarrow \mathsf{Open}(\boldsymbol{\rho}, 1)\end{array}\right] \approx_c M(h, \boldsymbol{\sigma})$$

   where $h \in \{0,1\}^N$ is such that $h_i = C(i)$ for all $i \in [N] = \{0,1\}^{\log N}$ and $M(h, \boldsymbol{\sigma})$ represents the actual measurement output in the basis defined by h.

Now, to state the soundness definition, we first specify some notation.

- $\mathcal{P}$: quantum register containing the public parameters pk

- $\mathcal{Y}$: quantum register containing the classical commitment string y
- $\mathcal{Z}$: quantum register containing the classical opening string z
- $\mathcal{I}$: the prover's initial state and internal working registers

A non-uniform cheating prover can be characterised as follows [BKL+22].

$$\mathsf{P}^* = (\boldsymbol{\rho_0}, U_{\mathsf{Com}^*}, U)$$

Definition 3.23 (Commit-and-Measure Protocol Soundness [BKL+22])**.** There exists an efficient classical algorithm $\mathsf{SimGen}(1^\lambda)$ and an efficient quantum algorithm $\mathsf{Extract}^{U,\boldsymbol{\rho}}(\mathsf{pk}, \mathsf{sk}, y)$ with black-box access to an attacker parameterized by a state $\boldsymbol{\rho}$ and a unitary U that takes as input classical strings $(\mathsf{pk}, \mathsf{sk}, y)$, and satisfies the following property. Consider any non-uniform QPT cheating prover $\mathsf{P}^* = (\boldsymbol{\rho_0}, U_{\mathsf{Com}^*}, U)$ that passes the test round with probability $1 - \mathsf{negl}(\lambda)$ for all $h \in \{0,1\}^N$. Then, for all $h \in \{0,1\}^N$ with circuit representation C, the following two distributions are computationally indistinguishable.

- **Real**
  1. Sample parameters $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, C)$.
  2. Run the attacker P^* on pk to obtain the classical commitment string y (i.e. apply U_{Com^*} and then measure the register containing y). Denote the post-measurement state as $\boldsymbol{\rho} \in D(\mathcal{Z} \otimes \mathcal{I})$, where $\mathcal{Z}$ corresponds to the registers that will eventually be measured to obtain the prover's final message, and $\mathcal{I}$ contains all of the other internal registers of the prover.
  3. Apply the prover's attack unitary U. This yields the state $\boldsymbol{\rho'} := U\boldsymbol{\rho}_{\mathcal{Z},\mathcal{I}}U^\dagger$. Measure the $\mathcal{Z}$ register of $\boldsymbol{\rho'}$ in the Hadamard basis to obtain the prover's opening string z.
  4. Compute $m \leftarrow \mathsf{Out}(\mathsf{sk}, (y, z))$ and output m.
- **Sim**
  1. Sample parameters $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{SimGen}(1^\lambda)$.
  2. Run the attacker P^* on pk to obtain a classical commitment string y (i.e. apply U_{Com^*} and then measure the register containing y). Denote the post-measurement state as $\boldsymbol{\rho} \in D(\mathcal{Z} \otimes \mathcal{I})$.
  3. Run $\mathsf{Extract}^{U,\boldsymbol{\rho}}(\mathsf{pk}, \mathsf{sk}, y) \to \boldsymbol{\tau}$ to obtain an $N-$qubit state $\boldsymbol{\tau}$.
  4. Measure each qubit of $\boldsymbol{\tau}$ according to the bases specified by $h \in \{0,1\}^N$ (i.e., qubit i is measured in the Hadamard basis if $h_i = 1$ and the standard basis if $h_i = 0$) and output the result.

3.9.5 Quantum Commit-Challenge-Response Protocols

Definition 3.24 (Quantum Commit-Challenge-Response Protocols [BKL+22])**.** Consider any *commit-challenge-measure* protocol between a quantum prover P and a classical verifier V, with the following three phases.

- **Commit**: $P(1^\lambda)$ and $V(1^\lambda; r)$ engage in a (potentially interactive) commitment protocol, where r are the random coins used by V.

- **Challenge**: V samples a random bit $b \leftarrow \{0,1\}$ and sends it to P.
- **Response**: P computes a (classical) response z and sends it to V.

After receiving the response, V accepts or rejects the execution.

We now describe some notations that are relevant for further definitions and discussion.

- Consider any non-uniform QPT prover P^* and let $\left|\psi^{P^*}_{\lambda,r}\right\rangle_{\mathcal{A},\mathcal{C}}$ be the (purified) state of the prover after interacting with $V(1^\lambda; r)$ in the commit phase, where $\mathcal{C}$ holds the (classical) prover messages output during this phase, and $\mathcal{A}$ holds the remaining state.
- The remaining strategy of the prover can be descrived by family of unitaries $\left\{U^{P^*}_{\lambda,0}, U^{P^*}_{\lambda,1}\right\}_{\lambda\in\mathbb{N}}$, where $U^{P^*}_{\lambda,0}$ is applied to $\left|\psi^{P^*}_{\lambda,r}\right\rangle$ if challenge is 0 and $U^{P^*}_{\lambda,1}$ if challenge is 1. This step is followed by a measurement yielding z.
- Let $V_{\lambda,r,0}$ (resp. $V_{\lambda,r,1}$) denote the accept/reject predicate applied by the verifier to the prover messages when $b = 0$ (resp. $b = 1$), written as a projection to be applied to the registers holding the prover messages. We define the following projectors on $\mathcal{A}\otimes\mathcal{C}$:

$$\Pi^{P^*}_{\lambda,r,0} := U^{P^*\dagger}_{\lambda,0} V_{\lambda,r,0} U^{P^*}_{\lambda,0}$$
$$\Pi^{P^*}_{\lambda,r,1} := U^{P^*\dagger}_{\lambda,1} V_{\lambda,r,1} U^{P^*}_{\lambda,1}$$

Definition 3.25 (Computationally Orthogonal Projectors [BKL+22])**.** A *commit-challenge-response* protocol has *computationally orthogonal projectors* if for any QPT prover P^*,

$$\mathbb{E}_r\left[\left\langle\psi^{P^*}_{\lambda,r}\middle|\Pi^{P^*}_{\lambda,r,0}\Pi^{P^*}_{\lambda,r,1}\Pi^{P^*}_{\lambda,r,0}\middle|\psi^{P^*}_{\lambda,r}\right\rangle\right] = \mathsf{negl}(\lambda)$$

Lemma 3.26 ([ACGH20])**.** Consider a *commit-challenge-response* protocol with the following properties.

1. $V_{\lambda,r,0}$ does not depend on r (that is, it is publicly computable given the transcript).
2. For any P^*, if $\mathbb{E}_r\left[\left\langle\psi^{P^*}_{\lambda,r}\middle|\Pi^{P^*}_{\lambda,r,0}\middle|\psi^{P^*}_{\lambda,r}\right\rangle\right] = 1 - \mathsf{negl}(\lambda)$, then $\mathbb{E}_r\left[\left\langle\psi^{P^*}_{\lambda,r}\middle|\Pi^{P^*}_{\lambda,r,1}\middle|\psi^{P^*}_{\lambda,r}\right\rangle\right] = \mathsf{negl}(\lambda)$.

Then, the protocol has *computationally orthogonal projectors*.

Theorem 3.27 ([ACGH20])**.** Consider the λ-fold parallel repetition of any *commit-challenge-response* protocol with computationally orthogonal projectors. The probability that the verifier accepts all λ parallel repetitions of the protocol is $\mathsf{negl}(\lambda)$.

3.9.6 The local Hamiltonian Problem

Definition 3.28 (2-local ZX-Hamiltonian Problem [ACGH20, BL08, CM16])**.** The 2-local ZX-Hamiltonian promise problem $\mathrm{zx} = (\mathrm{zx}_{yes}, \mathrm{zx}_{no})$, with parameters $a, b \in \mathbb{R}$, $b > a$, and gap $(b-a) > \mathsf{poly}(n)^{-1}$, is defined as follows. An instance is a local Hamiltonian

$$H = \sum_{i<j} J_{ij}(X_iX_j + Z_iZ_j)$$

where each J_{ij} is a real number such that $\sum_{i<j}|J_{ij}| = 1$ and each X_i (resp. Z_i) is a Pauli X (resp. Pauli Z) gate acting on the ith qubit. For $H \in \mathrm{zx}_{yes}$, the smallest eigenvalue of H is at most a, while if $H \in \mathrm{zx}_{no}$, the smallest eigenvalue of H is at least b.

3.9.7 Non-Interactive Post-Hoc Verification

An information-theoretic non-interactive QMA verification protocol is proposed in [FHM18] for verifying statements in all languages $\mathcal{L} = (\mathcal{L}_{\text{yes}}, \mathcal{L}_{\text{no}}) \in \mathsf{QMA}$. Consider $\text{zx}_{a,b}$, the corresponding instance of the 2-local Hamiltonian Problem.

Definition 3.29 (FHM Protocol [FHM18, ACGH20])**.** The FHM Protocol consists of a QPT algorithm P and a PPT algorithm V such that for a polynomial $\ell(\lambda)$ in the time $T(\lambda)$ required to verify instances of size λ, we have the following.

- $\mathsf{P}(x, |\phi\rangle) \to |\pi\rangle$: A QPT algorithm that uses the instance x and the witness state $|\phi\rangle$ to produce the $\ell(\lambda)$ qubit state $|\pi\rangle$ which functions as a proof for $x \in \mathcal{L}_{\text{yes}}$.
- $\mathsf{V}(x, S, m) \to \{0, 1\}$: A PPT algorithm that uses a term S of the Hamiltonian (sampled using a particular distribution), corresponding measurement results m to either accept or reject the instance at hand.

We direct the reader to [ACGH20] for complete details. A high-level execution of this protocol proceeds as follows. The prover prepares the ground state of the 2-local Hamiltonian at hand using the P algorithm and sends it to the verifier. The verifier samples a Hamiltonian term S from a particular problem-dependent distribution π. Subsequently, the verifier creates a basis string h such that if the observable for i-th qubit in S is Z (resp. X), then h_i is 0 (resp. 1). The received state $|\pi\rangle$ is then measured in the basis determined by h to get the measurement outcome m. Finally, the verifier outputs the result of $\mathsf{V}(x, S, m)$. This protocol has an inverse polynomial error. Therefore, we make use of an r-fold parallel repetition to achieve negligible soundness error when $r = \omega(\frac{\log \lambda}{(b-a)^2})$. Moreover, the above protocol uses instance-dependent basis string h; the protocol we describe now also gets around this shortcoming by virtue of some additional repetitions.

Definition 3.30 (κ-fold FHM Protocol with Instance Independent Setup [ACGH20])**.** The protocol consists of the following algorithms.

- $\mathsf{P}_{\mathsf{FHM}}(x, |\phi\rangle^{\otimes\kappa}) \to \rho$: A QPT algorithm that uses the instance x and the witness state $|\phi\rangle$ to produce the $\ell(\lambda)$ qubit state ρ (κ copies of the ground state $|\pi\rangle$) which functions as a proof for $x \in \mathcal{L}_{\text{yes}}$.
- $\mathsf{V}_{\mathsf{FHM}}(x, m, h) \to \{0, 1\}$: A PPT algorithm that uses the measurement string m of qubits of ρ to either accept or reject the instance at hand. Internally, this algorithm works as follows.
  1. Sample κ terms $S_1, \ldots, S_\kappa \leftarrow \pi$.
  2. Parse $(h_1, \ldots, h_\kappa) = h$ and $(m_1, \ldots, m_\kappa) = m$.
  3. Form a subset $A \subset [\kappa]$ of consistent copies. A copy $i \in A$ means h_i and S_i are consistent w.r.t. observables. Note that since we are dealing with 2-local Hamiltonians, about $\kappa/4$ copies are expected to be consistent for large enough κ.
  4. For each copy $i \in A$, the we set $v_i := \mathsf{V}(x, S_i, m_i)$.
  5. Output 1 if $\sum v_i \geq (1 - a - b)|A|/4$ otherwise 0.

The protocol proceeds as follows.

- **Setup**: The verifier samples κ random basis strings $h_1, \ldots, h_\kappa \leftarrow \{0, 1\}^\kappa$ uniformly.

- **Round 1**: The prover creates $\rho \leftarrow \mathsf{P}_{\mathsf{FHM}}(x, |\phi\rangle^{\otimes\kappa})$ and sends ρ to the verifier.
- **Round 2**: The verifier measures qubits of ρ in the basis defined by the string $h = (h_1, \ldots, h_\kappa)$ to get a measurement string $m = (m_1, \ldots, m_\kappa)$. It then outputs the result of $\mathsf{V}_{\mathsf{FHM}}(x, m, h)$.

Remark 3.31. As discussed in the above definition, for large enough κ, $|A| = \kappa/4$. Moreover, we have already established that to achieve negligible soundness error, we require $|A| = r = \omega(\frac{\log\lambda}{(b-a)^2})$. Consequently, $\kappa = 4 \times \omega(\frac{\log\lambda}{(b-a)^2})$. In other words, for Instance Independent Setup, we require a constant number of repetitions of the r-fold FHM protocol.

Definition 3.32 (κ-fold FHM Protocol Completeness and Soundness [FHM18, ACGH20, BKL+22])**.** The FHM protocol has the following security properties.

- **Completeness**: For all $x \in \mathcal{L}_{\text{yes}}$, for all valid witnesses $|\phi\rangle \in \mathcal{R}_{\mathcal{L}}(x)$ and for all basis choices $h \in \{0,1\}^{\ell(\lambda)}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that the following holds.

$$\Pr\left[\mathsf{V}_{\mathsf{FHM}}(x, m, h) = 1 : \boldsymbol{\rho} \leftarrow \mathsf{P}_{\mathsf{FHM}}(x, |\phi\rangle^{\otimes\kappa})\right] \geq 1 - \mathsf{negl}(\lambda)$$

  where m is the result of measuring qubits of $\boldsymbol{\rho}$ in the basis defined by h.

- **Soundness**: For every quantum adversary $\mathcal{A}$, for all $x \in \mathcal{L}_{\text{no}}$ and for all basis choices $h \in \{0,1\}^{\ell(\lambda)}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that the following holds.

$$\Pr\left[\mathsf{V}_{\mathsf{FHM}}(x, m, h) = 1 : \boldsymbol{\rho}^* \leftarrow \mathcal{A}(x)\right] \leq \mathsf{negl}(\lambda)$$

  where m is the result of measuring qubits of $\boldsymbol{\rho}^*$ in the basis defined by h.

4 Interactive Batch Arguments for QMA

In this section, we present a six-round (five-round with setup) interactive batch argument system for QMA from the post-quantum hardness of learning with errors. We note that this is a designated-verifier BARG. More formally, the result can be stated as follows.

Theorem 4.1 (Designated Verifier Interactive Batch Arguments for QMA)**.** Assuming the post-quantum hardness of the learning with errors problem (LWE), there is a designated verifier **6-round** interactive argument system for any batch of k T-time quantum computations on inputs $x_1, \ldots, x_k$ and corresponding $N = \mathsf{poly}(T)$ qubit witnesses $|\phi_1\rangle, \ldots, |\phi_k\rangle$, where

1. The Prover P is a QPT algorithm that runs in time $\mathsf{poly}(k, N)$
2. The Verifier V is a PPT algorithm that runs in time $\mathsf{poly}(\log k, N)$
3. The length of each message in the protocol is bounded by $\mathsf{poly}(\log k, N)$

The Interactive argument satisfies the BARG completeness and semi-adaptive computational soundness as defined in Section 3.9.3

4.1 Construction

We start with the BARG construction in [BKL+22] which makes use of state preserving arguments of knowledge. The insight here is that these arguments of knowledge can be replaced with seBARG to achieve a reduction in the number of rounds down to 6.

Ingredients

1. Somewhere Extractable BARG or seBARG(Setup, Prove, Verify, Extract)
2. Hash Family with Succinct Local Openings HT(Gen, Hash, Open, Verify)
3. Commit-and-Measure Protocol Maha(Gen, Com, Open, Test, Out) as defined in definition 3.21.
4. The κ-fold FHM post-hoc verification protocol $(\mathsf{P}_{\mathsf{FHM}}, \mathsf{V}_{\mathsf{FHM}})$ as defined in definition 3.30 where κ is a polynomial in instance length.

To simplify the presentation, we make use of an instance verification predicate defined as follows.

Definition 4.2 (Instance Verification Circuit (VerifyInstance))**.** The circuit $\mathsf{VerifyInstance}(x, y, z, c, r)$ essentially runs the following operations using the input.

1. Sample a basis string h using randomness r.
2. Generate public and secret parameters for the commit-and-measure protocol.
$$(\mathsf{mh.pk}, \mathsf{mh.sk}) = \mathsf{Maha.Gen}(1^\lambda, h; r)$$
3. If $c = 0$, output the result of $\mathsf{Maha.Test}(\mathsf{pk}, y, z)$.
4. If $c = 1$, perform the following and output res.
$$\begin{aligned} u &= \mathsf{Maha.Out}(\mathsf{mh.sk}, y, z) \\ \mathsf{res} &= \mathsf{V}_{\mathsf{FHM}}(x, u, h) \end{aligned}$$

We make use of the following languages.

Definition 4.3 (Knowledge Language)**.** A Knowledge Language $\mathcal{G}_{\mathsf{hk}}$ is defined as follows.

$$\mathcal{G}_{\mathsf{hk}} = \{(i, \alpha) \mid \exists z, \mathsf{op}_z \text{ such that } \mathsf{HT.Verify}(\mathsf{hk}, \alpha, i, z, \mathsf{op}_z) = 1\}$$

Additionally, let $\mathcal{R}_{\mathcal{G}}$ denote a relation s.t $\mathcal{R}_{\mathcal{G}}(i, \alpha) = 1$ iff $(i, \alpha) \in \mathcal{G}_{\mathsf{hk}}$. Let the corresponding circuit encoding this relation be $\mathcal{C}_{\mathcal{G}}$.

Definition 4.4 (Knowledge Language with Instance Verification)**.** A Knowledge Language with Instance Verification $\mathcal{K}_{\mathsf{hk}}$ is defined as follows.

$$\begin{aligned} \mathcal{K}_{\mathsf{hk}} = \{(i, x, \alpha, \beta, c, r) \mid & \exists y, \mathsf{op}_y \text{ s.t. } \mathsf{HT.Verify}(\mathsf{hk}, \alpha, i, y, \mathsf{op}_y) = 1 \\ & \wedge \ \exists z, \mathsf{op}_z \text{ s.t. } \mathsf{HT.Verify}(\mathsf{hk}, \beta, i, z, \mathsf{op}_z) = 1 \\ & \wedge \ \mathsf{VerifyInstance}(x, y, z, c, r) = 1\} \end{aligned}$$

Additionally, let $\mathcal{R}_{\mathcal{K}}$ denote a relation s.t $\mathcal{R}_{\mathcal{K}}(i, x, \alpha, \beta, c, r) = 1$ iff $(i, x, \alpha, \beta, c, r) \in \mathcal{K}_{\mathsf{hk}}$. Let the corresponding circuit encoding this relation be $\mathcal{C}_{\mathcal{K}}$.

Protocol Specification

- $\mathsf{Setup}(1^\lambda, 1^k, 1^N, j^*) \to (\mathsf{crs}, \mathsf{vk})$
  - Setup the commit-and-measure protocol Maha. Sample randomness $r \xleftarrow{\$} \{0,1\}^\lambda$ and subsequently sample the basis string h using the randomness r.
  $$(\mathsf{mh.pk}, \mathsf{mh.sk}) = \mathsf{Maha.Gen}(1^\lambda; r)$$
  - Setup HT as follows.
  $$\mathsf{ht.hk} \leftarrow \mathsf{HT.Gen}(1^k, 1^\lambda)$$
  - Generate $(\mathsf{sB.crs}_i, \mathsf{sB.td}_i)$ corresponding to seBARG_i for $i \in [2]$ and any random index $j \in [k]$ as follows.
  $$(\mathsf{sB.crs}_1, \mathsf{sB.td}_1) \leftarrow \mathsf{seBARG}_1.\mathsf{Setup}(1^\lambda, 1^k, 1^{|\mathcal{C}_\mathcal{G}|}, j)$$
  $$(\mathsf{sB.crs}_2, \mathsf{sB.td}_2) \leftarrow \mathsf{seBARG}_2.\mathsf{Setup}(1^\lambda, 1^k, 1^{|\mathcal{C}_\mathcal{K}|}, j)$$
  - Outputs the following.
  $$\mathsf{crs} = (\mathsf{mh.pk}, \mathsf{ht.hk}, \mathsf{sB.crs}_1, \mathsf{sB.crs}_2)$$
  $$\mathsf{vk} = (\mathsf{mh.sk}, r)$$
- $\mathsf{Prove}(\mathsf{crs}, \{x_j\}_{j\in[k]}, \{|\phi_j\rangle^{\otimes\kappa}\}_{j\in[k]})$: Here, the quantum state $|\phi_i\rangle$ is the quantum witness to the statement $x_i \in \mathcal{L}_{yes}$ for $\mathcal{L} \in \mathsf{QMA}$.
  - **Round 1**: Perform the following and send $(\hat{y}, \pi_y)$ to the verifier.
  $$\begin{aligned}(\mathsf{sB.crs}_1, \mathsf{ht.hk}, \mathsf{mh.pk}) &= \mathsf{crs}\\ \forall j,\ |\psi_j\rangle &\leftarrow \mathsf{P}_{\mathsf{FHM}}(x_j, |\phi_j\rangle^{\otimes\kappa})\\ \forall j,\ (y_j, |\mathsf{st}_j\rangle) &\leftarrow \mathsf{Maha.Com}(\mathsf{mh.pk}, |\psi_j\rangle)\\ \hat{y} &= \mathsf{HT.Hash}(\mathsf{ht.hk}, \{y_j\}_{j\in[k]})\\ \forall j,\ \mathsf{op}_j^{\hat{y}} &= \mathsf{HT.Open}(\mathsf{ht.hk}, \{y_j\}_{j\in[k]}, j)\\ \pi_y &\leftarrow \mathsf{seBARG}_1.\mathsf{Prove}(\mathsf{sB.crs}_1, \{(j, \hat{y})\}_{j\in[k]}, \{y_j, \mathsf{op}_j^{\hat{y}}\}_{j\in[k]})\end{aligned}$$
  - **Round 3**: Receive a bit $c \in \{0,1\}$. Perform the following and send $(\hat{z}, \pi_z)$ to the verifier.
  $$\begin{aligned}\forall j,\ z_j &\leftarrow \mathsf{Maha.Open}(|\mathsf{st}_j\rangle, c)\\ \hat{z} &= \mathsf{HT.Hash}(\mathsf{ht.hk}, \{z_j\}_{j\in[k]})\\ \forall j,\ \mathsf{op}_j^{\hat{z}} &= \mathsf{HT.Open}(\mathsf{ht.hk}, \{z_j\}_{j\in[k]}, j)\\ \pi_z &\leftarrow \mathsf{seBARG}_1.\mathsf{Prove}(\mathsf{sB.crs}_1, \{(j, \hat{z})\}_{j\in[k]}, \{z_j, \mathsf{op}_j^{\hat{z}}\}_{j\in[k]})\end{aligned}$$
  - **Round 5**: Receive a string r. Parse $\mathsf{sB.crs}_2$ from crs. Perform the following and send π_f to the verifier.
  $$\begin{aligned}\{w_j\}_{j\in[k]} &= \{(y_j, \mathsf{op}_j^{\hat{y}}, z_j, \mathsf{op}_j^{\hat{z}})\}_{j\in[k]}\\ \pi_f &\leftarrow \mathsf{seBARG}_2.\mathsf{Prove}(\mathsf{sB.crs}_2, \{(j, x_j, \hat{y}, \hat{z}, c, r)\}_{j\in[k]}, \{w_j\}_{j\in[k]})\end{aligned}$$

- $\mathsf{Verify}(\mathsf{crs}, \mathsf{vk}, \{x_j\}_{j\in[k]}, \pi_y, \pi_z, \pi_f)$: Works as follows under normal (honest) execution.
  - **Round 2**: Receive $(\hat{y}, \pi_y)$. Parse $\mathsf{sB.crs}_1$ from crs. Check if
    $$\mathsf{seBARG}_1.\mathsf{Verify}(\mathsf{sB.crs}_1, \{(j, \hat{y})\}_{j\in[k]}, \pi_y) = 1$$
    If yes, sample bit c and send it to the prover. If no, reject.
  - **Round 4**: Receive $(\hat{z}, \pi_z)$ from the prover. Check if
    $$\mathsf{seBARG}_1.\mathsf{Verify}(\mathsf{sB.crs}_1, \{(j, \hat{z})\}_{j\in[k]}, \pi_z) = 1$$
    If yes, parse vk to get r and send r to the prover. Otherwise, reject.
  - **Final Verification**: Receive π_f from the prover. Parse $\mathsf{sB.crs}_2$ from crs. Accept if
    $$\mathsf{seBARG}_2.\mathsf{Verify}(\mathsf{sB.crs}_2, \{(j, x_j, \hat{y}, \hat{z}, c, r)\}_{j\in[k]}, \pi_f) = 1$$
    Otherwise, reject.

Remark 4.5. We use $\mathfrak{P}$ to denote the above protocol and $\mathfrak{P}^\lambda$ to denote the λ-fold repetition of $\mathfrak{P}$.

4.2 Security Analysis

Completeness Completeness of $\mathfrak{P}$ follows immediately from the completeness properties of the underlying building blocks. If the instances $x_i \in \mathcal{L}_{\text{yes}}$ for all $i \in [k]$, we must have, by the completeness properties of the Commit and Measure protocol (Definition 3.22) and the FHM protocol (Definition 3.32), that $\mathsf{VerifyInstance}(x_i, \cdot, \cdot, \cdot, \cdot) = 1$ with non-negligible probability. By the completeness of seBARGs (Definition 3.6), we must have the proofs π_y, π_z, π_f always accepted for their respective languages $\mathcal{G}_{\mathsf{hk}}$ and $\mathcal{K}_{\mathsf{hk}}$. Therefore, the overall protocol accepts with a non-negligible probability if $x_i \in \mathcal{L}_{\text{yes}}$ for all $i \in [k]$. Furthermore, completeness of $\mathfrak{P}^\lambda$ then follows using a union-bound argument.

Round Complexity We observe that $\mathfrak{P}$ and (therefore) $\mathfrak{P}^\lambda$ both have **5 rounds** with setup. Alternatively, the verifier can do the setup and crs can be sent to the prover as the verifier's first message to get a **6 round** interactive BARG for QMA.

Soundness The protocol $\mathfrak{P}^\lambda$ satisfies the notion of *semi-adaptive computational soundness* as defined in Definition 3.20. We now provide the soundness proof.

Theorem 4.6. Assuming the soundness of the Commit-and-Measure protocol (Definition 3.23) and the FHM protocol (Definition 3.32), collision resistance w.r.t. opening of the hash tree family HT, and semi-adaptive somewhere computational soundness of seBARG, the protocol $\mathfrak{P}^\lambda$ satisfies semi-adaptive somewhere computational soundness as defined in definition 3.20.

Proof. The proof strategy is as follows. We note that $\mathfrak{P}$ is a *Commit-Challenge-Response* protocol as per Definition 3.24. For semi-adaptive, we first receive an index $i^* \in [k]$ from a QPT attacker $\mathcal{A}$. We begin by setting a bunch of hybrid experiments and show that the attacker's advantage has changed negligibly. Moreover, we note that the final hybrid satisfies the conditions of Lemma 3.26 through the soundness properties of the Commit and Measure protocol and FHM protocol. Thus, a parallel repetition of the final hybrid must have negligible soundness through Theorem 3.27. Since the protocol $\mathfrak{P}$ and the final hybrid have negligibly close advantage, it must be the case that a λ-fold repetition of $\mathfrak{P}$ must have negligible soundness.

Hybrid 0: Normal execution of the protocol

Hybrid 1: Same as previous execution except that we now enable extraction on index $i^\star$ (instead of a random index j)

$$(\mathsf{sB.crs}_1^\star, \mathsf{sB.td}_1^\star) \leftarrow \mathsf{seBARG}_1.\mathsf{Setup}(1^\lambda, 1^k, 1^{|\mathcal{C}_\mathcal{G}|}, i^\star)$$
$$(\mathsf{sB.crs}_2^\star, \mathsf{sB.td}_2^\star) \leftarrow \mathsf{seBARG}_2.\mathsf{Setup}(1^\lambda, 1^k, 1^{|\mathcal{C}_\mathcal{K}|}, i^\star)$$

Hybrid 2: We now update the protocol by making the following extractions at appropriate places on the verifier's end.

$$y'_{i^\star}, \mathsf{op}'^{\hat{y}}_{i^\star} \leftarrow \mathsf{seBARG}_1.\mathsf{Extract}(\mathsf{sB.td}_1^\star, \{(i, \hat{y})\}_{i\in[k]}, \pi_y)$$
$$z'_{i^\star}, \mathsf{op}'^{\hat{z}}_{i^\star} \leftarrow \mathsf{seBARG}_1.\mathsf{Extract}(\mathsf{sB.td}_1^\star, \{(i, \hat{z})\}_{i\in[k]}, \pi_z)$$
$$y^\star_{i^\star}, z^\star_{i^\star}, \mathsf{op}^{\hat{y}}_{i^\star}, \mathsf{op}^{\hat{z}}_{i^\star} \leftarrow \mathsf{seBARG}_2.\mathsf{Extract}(\mathsf{sB.td}_2^\star, \{(i, x_i, \hat{y}, \hat{z}, c, r)\}_{i\in[k]}, \pi_f)$$

In each round, reject if the extracted witnesses aren't valid by checking

$$\mathsf{HT.Verify}(\mathsf{ht.hk}, \hat{y}, i^\star, y'_{i^\star}, \mathsf{op}'^{\hat{y}}_{i^\star}) = \mathsf{HT.Verify}(\mathsf{ht.hk}, \hat{y}, i^\star, y^\star_{i^\star}, \mathsf{op}^{\hat{y}}_{i^\star}) = 1$$
$$\mathsf{HT.Verify}(\mathsf{ht.hk}, \hat{z}, i^\star, z'_{i^\star}, \mathsf{op}'^{\hat{z}}_{i^\star}) = \mathsf{HT.Verify}(\mathsf{ht.hk}, \hat{z}, i^\star, z^\star_{i^\star}, \mathsf{op}^{\hat{z}}_{i^\star}) = 1$$
$$\mathsf{VerifyInstance}(x_{i^\star}, y'_{i^\star}, z'_{i^\star}, c, r) = \mathsf{VerifyInstance}(x_{i^\star}, y^\star_{i^\star}, z^\star_{i^\star}, c, r) = 1$$

Hybrid 3: Same as previous execution except that now we add the following additional check in the final round of the protocol. We reject if the following are not true.

$$y^\star_{i^\star} = y'_{i^\star}, \qquad z^\star_{i^\star} = z'_{i^\star}$$

We now prove that the attacker's advantage changes negligibly through the sequence of hybrids assuming the soundness properties of the cryptographic building blocks being used. Subsequently, we show that **Hybrid 3** is a Commit-Challenge-Response protocol that has orthogonal projectors. We then show that since the attacker's advantage has changed negligibly from **Hybrid 0** to **Hybrid 3**, $\mathfrak{P}^\lambda$ cannot have non-negligible soundness. To begin, let us assume that the attacker's advantage in breaking the semi-adaptive somewhere computational soundness property of **Hybrid 0** is $\varepsilon(\lambda)$. We wish to show that $\varepsilon(\lambda)$ is a negligible function.

Lemma 4.7. Assuming seBARG satisfies the index hiding property, for every QPT attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid 0** and **Hybrid 1**.

Proof. Let us assume that the attacker $\mathcal{A}$ in **Hybrid 1** has a non-negligible advantage over an attacker in **Hybrid 0**. We can construct a reduction algorithm $\mathcal{B}$ to break the index-hiding property of seBARGs using $\mathcal{A}$. Let $\mathcal{C}$ be the challenger for the index-hiding property for seBARG_1. We perform the following operations

$$i^\star \leftarrow \mathcal{A}(1^\lambda, 1^k), \quad i \leftarrow \mathcal{B}(1^\lambda, 1^k)$$
$$(\mathsf{crs}') \leftarrow \mathcal{C}(i, i^\star)$$
$$(\pi'_y, \pi'_z) \leftarrow \mathcal{A}(\mathsf{crs}')$$

We observe that if the index i was used to generate the crs', we are operating in **Hybrid 0**. If the proofs (π'_y, π'_z) are accepted, then $\mathcal{B}$ can guess that the crs' was generated using index i^* rather than i, which contradicts the index-hiding property of seBARG (Definition 3.6). We can extend the same proof to show that the attacker cannot distinguish the crs generated for index i^* for seBARG_2. Therefore, there exists a negligible function $\mu_1(\lambda)$ s.t the advantage of $\mathcal{A}$ in **Hybrid 1** is $\varepsilon(\lambda) - \mu_1(\lambda)$. □

Lemma 4.8. Assuming seBARG satisfies the somewhere argument of knowledge property and HT satisfies opening completeness, for every QPT attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid 1** and **Hybrid 2**.

Proof. For contradiction, assume there is a non-negligible difference in the attacker's advantages in the two hybrids. We can design a reduction algorithm $\mathcal{B}$ that uses the QPT attacker $\mathcal{A}$ to break the **Somewhere Argument of Knowledge** property of seBARGs. Let $\mathcal{C}$ be the seBARG challenger for the language $\mathcal{G}_{\mathsf{hk}}$. We perform the following operations:

$$\begin{aligned}(\mathsf{sB.crs}^\star, \mathsf{sB.td}^\star) &\leftarrow \mathcal{C}(i^\star)\\ (\hat{y}^\star, \pi_y^\star) &\leftarrow \mathcal{A}(\mathsf{crs}^\star)\\ (y_{i^\star}^\star, \mathsf{op}_{i^\star}^{\hat{y}^\star}) &\leftarrow \mathsf{seBARG.Extract}(\mathsf{sB.td}^\star, i^\star, \pi_y^\star)\end{aligned}$$

$\mathcal{B}$ forwards the value of $\hat{y}^\star$ to $\mathcal{C}$ along with π_y. By the definition of $\mathcal{G}_{\mathsf{hk}}$, the Challenger $\mathcal{C}$ performs the check

$$\mathsf{seBARG.Verify}(\mathsf{sB.crs}^\star, \{(i, \hat{y}^\star)\}_{i\in[k]}, \pi_y^\star) = 1 \ \wedge \ \mathsf{HT.Verify}(\mathsf{ht.hk}, \hat{y}^\star, i^\star, y_{i^\star}^\star, \mathsf{op}_{i^\star}^{\hat{y}^\star}) = 1$$

Observe that the solitary check for the validity of the proof π_y^* operates in **Hybrid 1**, while the full check operates in **Hybrid 2**. We also have $\mathcal{C}$ rejecting whenever $\mathcal{B}$ rejects. Assuming opening completeness of HT (Definition 3.1), we must have that $(y_{i^\star}^\star, \mathsf{op}_{i^\star}^{\hat{y}^\star})$ is not a valid witness to the language $\mathcal{G}_{\mathsf{hk}}$ whenever $\mathcal{B}$ rejects. This contradicts the **Somewhere Argument of Knowledge** property of seBARGs (Definition 3.6). The same proof can be extended to the proofs π_z and π_f. Therefore, there exists a negligible function $\mu_2(\lambda)$ s.t the advantage of $\mathcal{A}$ in **Hybrid 2** is $\varepsilon(\lambda) - \mu_1(\lambda) - \mu_2(\lambda)$. □

Lemma 4.9. Assuming the collision resistance of the hash function used in HT with respect to the openings and hashed values, for every QPT attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid 2** and **Hybrid 3**.

Proof. A non-negligible advantage across these two hybrids implies that one of $y_{i^\star}^\star \neq y'_{i^\star}$ or $z_{i^\star}^\star \neq z'_{i^\star}$ is true. This contradicts the collision resistance of the hash function used in HT with respect to the openings and values. Therefore, we can quantify the advantage gain of $\mathcal{A}$ using a negligible function $\mu_3(\lambda)$ as $\varepsilon(\lambda) - \mu_1(\lambda) - \mu_2(\lambda) - \mu_3(\lambda)$. □

We take a break from the hybrid discussions and show that the protocol in **Hybrid 3**, hence referred to as $\mathfrak{P}'$, must have computationally orthogonal projectors. This result is crucial for us to show that the starting protocol $\mathfrak{P}$ also satisfies this property and thus has negligible soundness (through Theorem 3.27).

Lemma 4.10. Assuming the soundness of the Commit-and-Measure Protocol (Maha) and the FHM protocol, **Hybrid 3** is a Commit-Challenge-Response protocol with computationally orthogonal projectors.

Proof. To show this, we consider the two conditions stated in lemma 3.26. Condition 1 follows from the structure of the VerifyInstance predicate. For Condition 2, we consider a non-uniform QPT attacker $\mathcal{A}^\star$ which passes the test round with probability $1 - \mathsf{negl}(\lambda)$. Let a reduction algorithm $\mathcal{B}$ for the instance x_{i^*} be given access to $\mathcal{A}^\star(|\rho_0\rangle, U_{\mathsf{Com}^*}, U)$. We let a challenger $\mathcal{C}$ generate the $(\mathsf{mh.pk}, \mathsf{mh.sk})$ pair and send $\mathsf{mh.pk}$ to $\mathcal{B}$. We run the following operations to get the **Sim** distribution

$$\begin{aligned}(\mathsf{mh.pk}, \mathsf{mh.sk}) &\leftarrow \mathcal{B}.\mathsf{SimGen}(1^\lambda)\\ y' &\leftarrow \mathcal{A}^\star(\mathsf{mh.pk})\\ |\tau\rangle &\leftarrow \mathsf{Extract}^{\mathcal{A}^\star}(\mathsf{mh.pk}, \mathsf{mh.sk}, y')\end{aligned}$$

Let m' be the measurement of $|\tau\rangle$ in the basis specified by $\mathsf{mh.pk}$. The reduction algorithm runs $\mathsf{V}_{\mathsf{FHM}}(x_{i^*}, m', h)$. For the **Real** distribution, the reduction algorithm $\mathcal{B}$ uses $\mathcal{A}^\star$ to output (y, z) using the $\mathsf{mh.pk}$ sent by the challenger $\mathcal{C}$. If the verifier accepts and the local FHM verification check doesn't, then $\mathcal{B}$ differentiates between the **Real** and **Sim** experiments which contradicts the soundness of the Commit and Measure Protocol as in Definition 3.23.

Based on the above, we conclude that there exists a state $|\psi'_{i^*}\rangle$ such that the probability with which the verifier accepts in the measurement round is $\mathsf{negl}(\lambda)$-close to the probability that $\mathsf{V}_{\mathsf{FHM}}$ accepts. Moreover, by the soundness of the FHM Protocol, since $x_{i^*} \in \mathcal{L}_{no}$ it must be the case that:

$$\Pr\left[\mathsf{V}_{\mathsf{FHM}}(x_{i^*}, m, h) = 1\right] = \mathsf{negl}(\lambda)$$

where m lies in the distribution of the measurement outcomes of $|\psi'_{i^*}\rangle$ corresponding to basis string $h = (h_1, \ldots, h_\kappa)$ consisting of basis choices for each copy. Consequently, we must have:

$$\mathbb{E}_r\left[\left\langle \psi^{P^*}_{\lambda,r} \middle| \Pi^{P^*}_{\lambda,r,1} \middle| \psi^{P^*}_{\lambda,r} \right\rangle\right] = \mathsf{negl}(\lambda)$$

In conclusion, we have shown that both conditions required for lemma 3.26 hold and as a result **Hybrid 3** has computationally orthogonal projectors. □

Lemma 4.11. The λ-fold repetition of **Hybrid 3**, i.e $(\mathfrak{P}')^\lambda$, has negligible soundness.

Proof. This follows from Theorem 3.27. □

Lemma 4.12. The λ-fold repetition of $\mathfrak{P}$, i.e $\mathfrak{P}^\lambda$, has negligible soundness.

Proof. We started with a QPT attacker $\mathcal{A}$ who has non-negligible soundness advantage $\varepsilon(\lambda)$. Through Lemma 4.7, Lemma 4.8, and Lemma 4.9 we have shown that for any QPT attacker $\mathcal{A}$ for instance $i^\star$ has at least $\varepsilon(\lambda) - \mathsf{negl}(\lambda)$ chance of success in breaking the soundness of $\mathfrak{P}'$, where,

$$\mathsf{negl}(\lambda) = (\mu_1 + \mu_2 + \mu_3)(\lambda)$$

A λ-fold independent parallel repetition of $\mathfrak{P}$ has advantage $(\varepsilon(\lambda))^\lambda$. Thus, the attacker $\mathcal{A}$ has the following advantage for the λ-fold parallel repetition of $\mathfrak{P}'$

$$\prod_\lambda (\varepsilon(\lambda) - \mathsf{negl}(\lambda)) \sim (\varepsilon(\lambda))^\lambda - \mathsf{negl}(\lambda)$$

which is non-negligible if $(\varepsilon(\lambda))^\lambda$ is non-negligible contradicting Lemma 4.11. Therefore, we must have $\varepsilon(\lambda) = \mathsf{negl}(\lambda)$. Therefore, we note that $\mathfrak{P}^\lambda$ satisfies the semi-adaptive soundness property for BARGs □

□

5 Public Coin Batch Arguments for QMA

In this section, we outline a public coin batch argument system for QMA. The protocol is a public coin except for the verifier's first message, i.e. the setup phase. This property, as we describe in Section 9, allows for round collapsing under the Quantum Random Oracle Model. More formally, this result can be stated as follows.

Theorem 5.1 (Designated Verifier Public Coin Interactive Batch Arguments for **QMA**)**.** Assuming the post-quantum hardness of the learning with errors problem (LWE) and post-quantum fully homomorphic encryption (FHE), there is a designated verifier public coin **4-round** interactive argument system for any batch of k T-time quantum computations on inputs $x_1, \ldots, x_k$ and corresponding $\mathsf{poly}(T)$-qubit witnesses $|\phi_1\rangle, \ldots, |\phi_k\rangle$, where

1. The Prover P is a QPT algorithm that runs in time $\mathsf{poly}(k, N)$
2. The Verifier V is a PPT algorithm that runs in time $\mathsf{poly}(\log k, N)$
3. The first message in the protocol has length $\mathsf{poly}(k, N)$ while all other messages have length $\mathsf{poly}(\log k, N)$.

The public coin protocol satisfies the BARG completeness and semi-adaptive somewhere computational soundness properties as defined in Section 3.9.3

5.1 FHE based AND Tree Construction

We now take a break from the BARG discussion and present three algorithms that builds an AND tree (ATree) using FHE ciphertexts. These algorithms are useful in ensuring all prover messages after **Round 1** (or Setup) of the protocol are succinct with respect to the batch size k. We first provide a definition that sets expectations from each of these algorithms. The high level idea is to bootstrap a FHE based binary tree using logical AND operations starting from ciphertexts encrypting booleans in the leaves. By decrypting the root, one can obtain the AND of all the input ciphertexts. Moreover, the tree structure, inspired from [Mer87], can be used to generate and verify succinct openings. Like in software development, we bundle each algorithm under the common name of ATree for a simplified presentation.

Definition 5.2 (FHE based AND Tree)**.** An FHE based AND tree ATree is a tuple (Build, Open, Verify) of algorithms defined as follows.

- $\mathsf{Build}(\mathsf{pk}, \mathsf{aux}, \mathsf{ct}) \to \mathsf{av}$: A deterministic poly-time algorithm that takes input n FHE ciphertexts $\mathsf{ct} = \{\mathsf{ct}_i\}_{i\in[n]}$, a set of FHE public keys pk, and some auxiliary information aux and computes the tree root av as the logical AND of all bits encoded by the input ciphertexts.
- $\mathsf{Open}(\mathsf{pk}, \mathsf{aux}, \mathsf{ct}, j) \to \mathsf{op}$: A deterministic poly time algorithm that outputs an opening op using a set of FHE public keys pk, some auxiliary information aux, the set of ciphertexts $\mathsf{ct} = \{\mathsf{ct}_i\}_{i\in[n]}$, and an index $j \in [n]$.
- $\mathsf{Verify}(\mathsf{pk}, \mathsf{aux}, \mathsf{av}, \mathsf{ct}_j, j, \mathsf{op}) \to \{0,1\}$: A deterministic poly-time algorithm that verifies that the opening op is valid with respect to (av, j) for some index $j \in [n]$ and ciphertext ct_j.

Definition 5.3 (ATree.Build procedure)**.** We define a procedure $\mathsf{ATree.Build}(\mathsf{pk}, \mathsf{aux}, \mathsf{ct} = \{\mathsf{ct}_i\}_{i\in[n]})$ that aims to compute the logical AND of all bits encoded under ct_i for all $i \in [n]$. The output is succinct w.r.t. n. Let ATree.Build perform the following computations:

1. Without loss of generality, assume $n = 2^h$ for some h. If not, pad the input with encryption of 1's to get to the next largest power of 2.
2. Consider a binary tree of height h whose vertices are $v \in \{0,1\}^{\leq h}$. The root of the binary tree is the empty string ϵ.
3. Parse $\mathsf{pk} = \{\mathsf{fhe.pk}_v\}$ for all binary strings $v \in \{0,1\}^{\leq h-1}$.
4. Parse $\mathsf{aux} = \{\mathsf{aux}_v = (\mathsf{aux}_{v,0}, \mathsf{aux}_{v,1})\}$ for all $v \in \{0,1\}^{\leq h-1}$.
5. For all $j \in [n]$, write j as a string of bits $v \in \{0,1\}^h$. Parse $\mathsf{ct} = \{\mathsf{ct}_v\}$
6. For $\ell = h$, set $\widehat{\mathsf{ct}}_v = \mathsf{ct}_v$.
7. For every $0 \leq \ell \leq h-1$, for every vertex $v \in \{0,1\}^\ell$, compute

$$\begin{aligned} \mathsf{aux}_v &= (\mathsf{aux}_{v,0}, \mathsf{aux}_{v,1}) \\ \widehat{\mathsf{ct}}_{v,\beta} &= \mathsf{FHE.Eval}\left(\mathsf{fhe.pk}_v, \mathsf{FHE.Dec}(\cdot, \widehat{\mathsf{ct}}_w), \mathsf{aux}_{v,\beta}\right), \forall \beta \in \{0,1\} \end{aligned}$$

   where $w = v\|\beta$ is appending β to v. Finally compute for the binary logical AND circuit C,

$$\widehat{\mathsf{ct}}_v = \mathsf{FHE.Eval}\left(\mathsf{fhe.pk}_v, C, (\widehat{\mathsf{ct}}_{v,0}, \widehat{\mathsf{ct}}_{v,1})\right)$$

8. Output $\mathsf{av} = \widehat{\mathsf{ct}}_\epsilon$ associated with the root of the tree.

Definition 5.4 (ATree.Open procedure)**.** We define a procedure $\mathsf{ATree.Open}(\mathsf{pk}, \mathsf{aux}, \mathsf{ct}, j)$ that produces an opening op to a ciphertext ct_j for some $j \in [n]$. Let ATree.Open perform the following operations:

1. For all $i \in [n]$, write i as a string of bits $v \in \{0,1\}^h$. Parse $\mathsf{ct} = \{\mathsf{ct}_v\}$
2. Let the binary representation of j be u, i.e $j \in \{0,1\}^h$.
3. For every $0 \leq \ell \leq h$, let v_ℓ be the vertex on the path from root ϵ to j at the level ℓ. Let v'_ℓ be the sibling node of v_ℓ.
4. Compute the tree as in ATree.Build and store all the ciphertexts $\widehat{\mathsf{ct}}_v$.
5. Output $\mathsf{op} = \left(\widehat{\mathsf{ct}}_u, \{\widehat{\mathsf{ct}}_{v'_\ell}\}_{0\leq\ell\leq h}\right)$

Definition 5.5 (ATree.Verify procedure)**.** We define a procedure $\mathsf{ATree.Verify}(\mathsf{pk}, \mathsf{aux}, \mathsf{av}, \mathsf{ct}_j, j, \mathsf{op})$ that determines whether the opening op is valid w.r.t. (av, j) for some $j \in [n]$. Let $\mathsf{ATree.Verify}$ perform the following computations:

1. Parse $\mathsf{pk} = \{\mathsf{fhe.pk}_v\}$ for all $v \in \{0,1\}^{\leq h-1}$.
2. Parse $\mathsf{aux} = \{\mathsf{aux}_v = (\mathsf{aux}_{v,0}, \mathsf{aux}_{v,1})\}$ for all $v \in \{0,1\}^{\leq h-1}$.
3. Parse $\mathsf{op} = \left(\widehat{\mathsf{ct}}^{\star}_u, \{\widehat{\mathsf{ct}}^{\star}_{v'_\ell}\}_{0 \leq \ell \leq h}\right)$
4. Accept if all of the following checks pass:
   (a) Let the binary representation of j be u. Check if $\widehat{\mathsf{ct}}^{\star}_u = \mathsf{ct}_u$. Let $\mathsf{ct}'_u = \widehat{\mathsf{ct}}^{\star}_u$.
   (b) For every $0 \leq \ell \leq h-1$, compute

$$\begin{aligned}
\mathsf{aux}_{v_\ell} &= (\mathsf{aux}_{v_{\ell,0}}, \mathsf{aux}_{v_{\ell,1}}) \\
\mathsf{ct}'_{v_\ell,0} &= \mathsf{FHE.Eval}\left(\mathsf{fhe.pk}_{v_\ell}, \mathsf{FHE.Dec}(\cdot, \mathsf{ct}'_{v_{\ell+1}}), \mathsf{aux}_{v_\ell,0}\right) \\
\mathsf{ct}'_{v_\ell,1} &= \mathsf{FHE.Eval}\left(\mathsf{fhe.pk}_{v_\ell}, \mathsf{FHE.Dec}(\cdot, \widehat{\mathsf{ct}}^{\star}_{v'_{\ell+1}}), \mathsf{aux}_{v_\ell,1}\right) \\
\mathsf{ct}'_{v_\ell} &= \mathsf{FHE.Eval}(\mathsf{fhe.pk}_{v_\ell}, C, \mathsf{ct}'_{v_\ell,0}, \mathsf{ct}'_{v_\ell,1})
\end{aligned}$$

   where C is the logical AND circuit. We continue this procedure until $\ell = 0$ at which point we have ct'_{v_0}.
   (c) Check if $\mathsf{ct}'_{v_0} = \mathsf{av}$.

5.2 Construction

Ingredients

1. Somewhere Extractable BARG or $\mathsf{seBARG}(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Extract})$
2. Fully Homomorphic Encryption or $\mathsf{FHE}(\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Enc}, \mathsf{Dec})$
3. Hash Family with Succinct Local Openings $\mathsf{HT}(\mathsf{Gen}, \mathsf{Hash}, \mathsf{Open}, \mathsf{Verify})$
4. The FHE based AND Tree $\mathsf{ATree}(\mathsf{Build}, \mathsf{Open}, \mathsf{Verify})$ as in Definition 5.2.
5. Commit-and-Measure Protocol $\mathsf{Maha}(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open}, \mathsf{Test}, \mathsf{Out})$ as defined in Definition 3.21.
6. The κ-fold FHM post-hoc verification protocol $(\mathsf{P}_{\mathsf{FHM}}, \mathsf{V}_{\mathsf{FHM}})$ as defined in Definition 3.30 where κ is a polynomial in instance length.

In addition to the *Knowledge Language* defined in Section 4, we make use of another extension of this notion. This can be defined as follows.

Definition 5.6 (FHM Knowledge Language)**.** An FHM knowledge language $\mathcal{G}$ with hard-coded parameters $\mathsf{pk}, \mathsf{hk}, \mathsf{ct}_r, \mathsf{aux}$ is defined as follows.

$$\begin{aligned}
\mathcal{G} = \{(i, x, \alpha, \beta, \gamma, c) \mid \quad & \exists\ y, \mathsf{op}_y \text{ s.t. } \mathsf{HT.Verify}(\mathsf{hk}, \alpha, i, y, \mathsf{op}_y) = 1 \\
\wedge \quad & \exists\ z, \mathsf{op}_z \text{ s.t. } \mathsf{HT.Verify}(\mathsf{hk}, \beta, i, z, \mathsf{op}_z) = 1 \\
\wedge \quad & \exists\ \mathsf{op}_{\mathsf{ct}} \text{ s.t. } \mathsf{ATree.Verify}(\mathsf{pk}, \mathsf{aux}, \gamma, \mathsf{ct}, i, \mathsf{op}_{\mathsf{ct}}) = 1 \\
\wedge \quad & \mathsf{ct} = \mathsf{FHE.Eval}(\mathsf{pk}, \mathsf{VerifyInstance}(x, y, z, c, \cdot), \mathsf{ct}_r)\}
\end{aligned}$$

Additionally, let $\mathcal{R}'_{\mathcal{G}}$ denote a relation s.t $\mathcal{R}'_{\mathcal{G}}(i, x, \alpha, \beta, \gamma, c) = 1$ iff $(i, x, \alpha, \beta, \gamma, c) \in \mathcal{G}$. Let the corresponding circuit encoding this relation be $\mathcal{C}'_{\mathcal{G}}$

Protocol Specification

- $\mathsf{Setup}(1^\lambda, 1^k, 1^N, j^*) \to (\mathsf{crs}, \mathsf{vk})$: The Setup algorithm generates a crs that grows as $\mathsf{poly}(\lambda, k, N)$. We use two indexing schemes: the standard integer indexing scheme and a binary string indexing scheme. The conversion between them, unless explicitly mentioned, is implied by the context.

  - Setup the commit-and-measure protocol Maha. For each $i \in [k]$, Sample randomness $r_i \xleftarrow{\$} \{0,1\}^\lambda$. Sample basis string h_i using randomness r_i.

    $$\forall i \in [k],\ (\mathsf{mh.pk}_i, \mathsf{mh.sk}_i) \leftarrow \mathsf{Maha.Gen}(1^\lambda, h_i; r_i)$$

  - Let the FHE keys be indexed by binary strings $v \in \{0,1\}^{\leq \log k}$, where $v = \epsilon$ represents the root of a binary tree. For all v, setup FHE as follows

    $$(\mathsf{fhe.pk}_v, \mathsf{fhe.sk}_v) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda)$$

    For all $i \in [k]$, use the binary representation of $u \in \{0,1\}^{\log k}$ and perform the following.

    $$\mathsf{ct}_{r_i} \leftarrow \mathsf{FHE.Enc}(\mathsf{fhe.pk}_i, r_i)$$

    Compute aux_v for all binary strings $v \in \{0,1\}^{\leq \log k - 1}$ as follows:

    $$\mathsf{aux}_v = \big(\mathsf{FHE.Enc}(\mathsf{fhe.pk}_v, \mathsf{fhe.sk}_{v\|0}), \mathsf{FHE.Enc}(\mathsf{fhe.pk}_v, \mathsf{fhe.sk}_{v\|1})\big)$$

  - Setup HT as follows.

    $$\mathsf{ht.hk} \leftarrow \mathsf{HT.Gen}(1^k, 1^\lambda)$$

  - Generate $\mathsf{sB.crs}_i$ for seBARG_i for $i \in [2]$ as follows for any $j \in [k]$

    $$(\mathsf{sB.crs}_1, \mathsf{sB.td}_1) \leftarrow \mathsf{seBARG}_1.\mathsf{Setup}(1^\lambda, 1^k, 1^{|\mathcal{C}_\mathcal{G}|}, j)$$
    $$(\mathsf{sB.crs}_2, \mathsf{sB.td}_2) \leftarrow \mathsf{seBARG}_2.\mathsf{Setup}(1^\lambda, 1^k, 1^{|\mathcal{C}'_\mathcal{G}|}, j)$$

  - Let $l = \log k$. Set outputs as follows.

    $$\begin{aligned}
    \mathbf{mh.pk} &= (\mathsf{mh.pk}_{0^l}, \ldots, \mathsf{mh.pk}_{1^l}) \\
    \mathbf{fhe.pk} &= (\mathsf{fhe.pk}_\epsilon, \ldots, \mathsf{fhe.pk}_{1^l}) \\
    \mathbf{aux} &= (\mathsf{aux}_\epsilon, \ldots, \mathsf{aux}_{1^{l-1}}) \\
    \mathbf{ct_r} &= (\mathsf{ct}_{r_1}, \ldots, \mathsf{ct}_{r_k}) \\
    \mathsf{crs} &= (\mathbf{mh.pk}, \mathbf{fhe.pk}, \mathbf{aux}, \mathsf{ht.hk}, \mathsf{sB.crs}_1, \mathsf{sB.crs}_2, \mathbf{ct_r}) \\
    \mathsf{vk} &= \mathsf{fhe.sk}_\epsilon
    \end{aligned}$$

- $\mathsf{Prove}(\mathsf{crs}, \{x_j\}_{j\in[k]}, \{|\phi_j\rangle^{\otimes\kappa}\}_{j\in[k]})$: Here, the quantum state $|\phi_i\rangle$ is the quantum witness to the statement $x_i \in \mathcal{L}_{yes}$ for $\mathcal{L} \in \mathsf{QMA}$.

– **Round 1**: Perform the following and send $(\hat{y}, \pi_y)$ to the verifier.

$$\begin{aligned}
(\mathsf{sB.crs}_1, \mathsf{ht.hk}, \mathbf{mh.pk}) &= \mathsf{crs} \\
(\mathsf{mh.pk}_1, \ldots, \mathsf{mh.pk}_k) &= \mathbf{mh.pk} \\
\forall j,\ |\psi_j\rangle &\leftarrow \mathsf{P}_{\mathsf{FHM}}(x_j, |\phi_j\rangle^{\otimes\kappa}) \\
\forall j,\ (y_j, |\mathsf{st}_j\rangle) &\leftarrow \mathsf{Maha.Com}(\mathsf{mh.pk}_j, |\psi_j\rangle) \\
\hat{y} &= \mathsf{HT.Hash}(\mathsf{ht.hk}, \{y_j\}_{j\in[k]}) \\
\forall j,\ \mathsf{op}_j^{\hat{y}} &= \mathsf{HT.Open}(\mathsf{ht.hk}, \{y_j\}_{j\in[k]}, j) \\
\pi_y &\leftarrow \mathsf{seBARG}_1.\mathsf{Prove}(\mathsf{sB.crs}_1, \{(j,\hat{y})\}_{j\in[k]}, \{y_j, \mathsf{op}_j^{\hat{y}}\}_{j\in[k]})
\end{aligned}$$

– **Round 3**: Receive bit $c \in \{0,1\}$. Parse $\mathsf{fhe.pk}$ to get the public keys at level $\ell = \log k$. Perform the following and send $(\mathsf{ct}^*, \pi_{\mathsf{ct}})$ to the verifier.

1. Parse the crs

$$\begin{aligned}
(\mathbf{fhe.pk}, \mathbf{aux}, \mathbf{ct}_{\boldsymbol{r}}, \mathsf{sB.crs}_2) &= \mathsf{crs} \\
(\mathsf{fhe.pk}_{0^\ell}, \mathsf{fhe.pk}_2, \ldots, \mathsf{fhe.pk}_{1^\ell}) &= \mathbf{fhe.pk} \\
(\mathsf{ct}_{r_1}, \mathsf{ct}_{r_2}, \ldots, \mathsf{ct}_{r_k}) &= \mathbf{ct}_{\boldsymbol{r}}
\end{aligned}$$

2. Run $\mathsf{Maha.Open}$

$$\forall j,\ z_j \leftarrow \mathsf{Maha.Open}(|\mathsf{st}_j\rangle, c)$$

3. Build ATree and compute the openings for each $j \in [k]$. Write all numbers j as binary strings v of length $\log k$.

$$\begin{aligned}
\mathsf{ct}_v &\leftarrow \mathsf{FHE.Eval}(\mathsf{fhe.pk}_v, \mathsf{VerifyInstance}(x_v, y_v, z_v, c, \cdot), \mathsf{ct}_{r_v}) \\
\mathsf{ct}^\star &= \mathsf{ATree.Build}(\mathsf{fhe.pk}, \mathsf{aux}, \{\mathsf{ct}_v\}) \\
\mathsf{op}_v^{\mathsf{ct}^\star} &= \mathsf{ATree.Open}(\mathsf{fhe.pk}, \mathsf{aux}, \{\mathsf{ct}_v\}, v))
\end{aligned}$$

4. Compute the proof π_{ct} as follows

$$\begin{aligned}
\{w_j\}_{j\in[k]} &= \{(y_j, \mathsf{op}_j^{\hat{y}}, z_j, \mathsf{op}_j^{\hat{z}}, \mathsf{ct}_j, \mathsf{op}_j^{\mathsf{ct}^\star})\}_{j\in[k]} \\
\pi_{\mathsf{ct}} &\leftarrow \mathsf{seBARG}_2.\mathsf{Prove}(\mathsf{sB.crs}_2, \{(j, x_j, \hat{y}, \hat{z}, \mathsf{ct}^\star, c)\}_{j\in[k]}, \{w_j\}_{j\in[k]})
\end{aligned}$$

- $\mathsf{Verify}(\mathsf{crs}, \mathsf{vk}, \{x_j\}_{j\in[k]}, \hat{y}, \pi_y, \pi_{\mathsf{ct}}, \mathsf{ct}^\star)$: Works as follows under normal (honest) execution.

– **Round 2**: Receive $(\hat{y}, \pi_y)$. Parse $\mathsf{sB.crs}_1$ from crs. Check if

$$\mathsf{seBARG}_1.\mathsf{Verify}(\mathsf{sB.crs}_1, \{(j,\hat{y})\}_{j\in[k]}, \pi_y) = 1$$

If yes, sample bit c and send it to the prover. If not, reject.

– **Final Verification**: Receive $(\mathsf{ct}^\star, \pi_{\mathsf{ct}})$ from the verifier. Parse $\mathsf{fhe.sk}_\epsilon$ from vk and $\mathsf{sB.crs}_2$ from crs. Accept if

$$\begin{aligned}
\mathsf{seBARG}_2.\mathsf{Verify}(\mathsf{sB.crs}_2, \{(j, x_j, \hat{y}, \hat{z}, \mathsf{ct}^\star, c)\}_{j\in[k]}, \pi_{\mathsf{ct}}) = 1 \wedge \\
\mathsf{FHE.Dec}(\mathsf{fhe.sk}_\epsilon, \mathsf{ct}^\star) = 1
\end{aligned}$$

Remark 5.7. We denote the above protocol as $\mathfrak{M}$ and $\mathfrak{M}^\lambda$ is a λ-fold repetition.

5.3 Security Analysis

Completeness Completeness of $\mathfrak{M}$ follows immediately from the completeness properties of the underlying building blocks. If the instances $x_i \in \mathcal{L}_{\text{yes}}$ for all $i \in [k]$, we must have, by the completeness properties of the Commit and Measure protocol (Definition 3.22) and the FHM protocol (Definition 3.32), that $\mathsf{VerifyInstance}(x_i, \cdot, \cdot, \cdot) = 1$ with non-negligible probability. By the correctness of the post quantum FHE-scheme (Definition 3.7), we must have that the leaves of the AND-tree be encryption of 1. By the correctness of $\mathsf{ATree.Build}$ (Definition 5.3), we must have the root $\mathsf{ct}^\star$ of the FHE based AND-tree be an encryption of 1. By the completeness of seBARGs (Definition 3.6), correctness of $\mathsf{ATree.Open}$ (Definition 5.4), and $\mathsf{ATree.Verify}$ (Definition 5.5), we must have the proofs π_y, π_{ct} always being accepted. Therefore, the overall protocol accepts with non-negligible probability if $x_i \in \mathcal{L}_{\text{yes}}$ for all $i \in [k]$. Furthermore, completeness of $\mathfrak{M}^\lambda$ then follows using a union bound argument.

Round Complexity We observe that $\mathfrak{M}$ and (therefore) $\mathfrak{M}^\lambda$ both have **3 rounds** with setup. Alternatively, the verifier can do the setup and the crs can be sent to the prover as the first message of the verifier to get a **4 round** Interactive **Public Coin** BARG for QMA.

Soundness The protocol $\mathfrak{M}^\lambda$ satisfies the notion of *semi-adaptive computational soundness* as defined in definition 3.20. We now provide the soundness proof.

Theorem 5.8. Assuming the soundness of the Commit-and-Measure protocol (Definition 3.23) and the FHM protocol (Definition 3.32), collision resistance w.r.t. opening of the hash tree family HT, semi-adaptive computational soundness of seBARG, and the semantic security of the post-quantum fully homomorphic encryption (FHE) scheme, the protocol $\mathfrak{M}^\lambda$ satisfies semi-adaptive soundness as defined in Definition 3.20.

Proof. We begin by setting a bunch of hybrid experiments and replacing the final round FHE decryption step by running the $\mathsf{VerifyInstance}$ predicate on the instance $i^\star$ chosen by the QPT attacker $\mathcal{A}$. Next, we show that the final hybrid must have computationally orthogonal projectors as defined in Definition 3.24. Finally, we claim that the λ-fold repetition of $\mathfrak{M}$ must have negligible soundness since the attacker gained negligible advantage to reach the final hybrid.

Hybrid 0: Normal execution of the protocol.

Hybrid 1: Same as previous execution except enable extraction on index $i^\star$ (instead of a random index j), i.e

$$\begin{aligned}(\mathsf{sB.crs}_1^\star, \mathsf{sB.td}_1^\star) &\leftarrow \mathsf{seBARG}_1.\mathsf{Setup}(1^\lambda, 1^k, 1^{|\mathcal{C}_{\mathcal{G}}|}, i^\star)\\ (\mathsf{sB.crs}_2^\star, \mathsf{sB.td}_2^\star) &\leftarrow \mathsf{seBARG}_2.\mathsf{Setup}(1^\lambda, 1^k, 1^{|\mathcal{C}'_{\mathcal{G}}|}, i^\star)\end{aligned}$$

Hybrid 2: We now update the protocol by making the following extractions at appropriate places on the verifier's end.

$$\begin{aligned}y'_{i^\star}, \mathsf{op}'^{\hat{y}}_{i^\star} &\leftarrow \mathsf{seBARG}_1.\mathsf{Extract}(\mathsf{sB.td}_1^\star, \{(i, \hat{y})\}_{i\in[k]}, \pi_y)\\ y^\star_{i^\star}, z^\star_{i^\star}, \mathsf{ct}^\star_{i^\star}, \mathsf{op}^{\hat{y}}_{i^\star}, \mathsf{op}^{\hat{z}}_{i^\star}, \mathsf{op}^{\mathsf{ct}^\star}_{i^\star} &\leftarrow \mathsf{seBARG}_2.\mathsf{Extract}(\mathsf{sB.td}_2^\star, \{(i, x_i, \hat{y}, \hat{z}, \mathsf{ct}^\star)\}_{i\in[k]}, \pi_f)\end{aligned}$$

In the final round, reject if the extracted witnesses aren't valid by checking the following condition.

$$\begin{aligned}\mathsf{HT.Verify}(\mathsf{ht.hk}, \hat{y}, i^\star, y'_{i^\star}, \mathsf{op}'^{\hat{y}}_{i^\star}) &= 1 \wedge \\ \mathsf{HT.Verify}(\mathsf{ht.hk}, \hat{y}, i^\star, y^\star_{i^\star}, \mathsf{op}^{\hat{y}}_{i^\star}) &= 1 \wedge \\ \mathsf{ATree.Verify}(\mathsf{fhe.pk}, \mathsf{aux}, \mathsf{ct}^\star, i^\star, \mathsf{ct}^\star_{i^\star}, \mathsf{op}^{\mathsf{ct}^\star}_{i^\star}) &= 1\end{aligned}$$

Hybrid 3: Same as previous execution except that now we add the following additional check in the final round of the protocol. We reject if the following is not true

$$y^\star_{i^\star} = y'_{i^\star}$$

Hybrid 4: This hybrid will also be referred as **Hybrid (4,0)**. Let u be the binary representation of $i^\star$. Parse

$$\left(\mathsf{ct}^\star_u, \{\mathsf{ct}^\star_{v'_\ell}\}_{0\le\ell\le\log k}\right) = \mathsf{op}^{\mathsf{ct}^\star}_{i^\star}$$

Perform the following in the final round along with the seBARG proof verifications

$$\mathsf{FHE.Dec}(\mathsf{fhe.sk}_\epsilon, \mathsf{ct}^\star) = 1 \ \wedge \ \mathsf{FHE.Dec}(\mathsf{fhe.sk}_{v'_0}, \mathsf{ct}^\star_{v'_0}) = 1$$

We introduce $\log k + 2$ intermediate hybrids where the challenger decrypts the ciphertexts received with the opening $\mathsf{op}^{\mathsf{ct}^\star}_{i^\star}$. Let the ℓth intermediate hybrid be **Hybrid (4, ℓ)**, where $0 \le \ell \le \log k + 1$. We define ℓ^{th} hybrid to be the same as $(\ell-1)^{\text{th}}$ hybrid but with the additional check

$$\mathsf{FHE.Dec}(\mathsf{fhe.sk}_{v'_\ell}, \mathsf{ct}^\star_{v'_\ell}) = 1$$

Thus, we have **Hybrid (4, $\log k$)** checking the following:

$$\mathsf{FHE.Dec}(\mathsf{fhe.sk}_\epsilon, \mathsf{ct}^\star) = 1 \ \wedge \ \ldots \ \wedge \ \mathsf{FHE.Dec}(\mathsf{fhe.sk}_u, \mathsf{ct}^\star_u) = 1$$

We define **Hybrid (4, $\log k + 1$)** as a hybrid where we drop the decryption of $\mathsf{ct}^\star$, i.e., we run the following check

$$\mathsf{FHE.Dec}(\mathsf{fhe.sk}_u, \mathsf{ct}^\star_u) = 1 \ \wedge \ \ldots \ \wedge \ \mathsf{FHE.Dec}(\mathsf{fhe.sk}_{v'_0}, \mathsf{ct}^\star_{v'_0}) = 1$$

Hybrid 5: We replace the FHE.Dec for index $i^\star$ by performing the check

$$\mathsf{VerifyInstance}(x_{i^\star}, y^\star_{i^\star}, z^\star_{i^\star}, c, r_{i^\star}) = 1$$

Hybrid 6: We now replace $\mathsf{ct}_{r_{i^\star}}$ in the crs with $\mathsf{ct}_0 = \mathsf{FHE.Enc}(\mathsf{fhe.pk}_{i^\star}, 0^\lambda)$ and let the protocol continue as in the previous hybrid.

We show that **Hybrid 6** is a Commit-Challenge-Response protocol that has orthogonal projectors. We then show that since the attacker's advantage has changed negligibly from **Hybrid 0** to **Hybrid 6**, $\mathfrak{M}^\lambda$ cannot have non-negligible soundness. To begin, let us assume that the attacker's advantage in breaking the semi-adaptive somewhere computational soundness property of **Hybrid 0** is $\varepsilon(\lambda)$. We wish to show that $\varepsilon(\lambda)$ is a negligible function.

Lemma 5.9. Assuming seBARG satisfies the index hiding property, for every QPT attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid 0** and **Hybrid 1**.

Proof. The proof of this lemma follows from Lemma 4.7. We can find a negligible function $\mu_1(\lambda)$ s.t the advantage of $\mathcal{A}$ in **Hybrid 1** is $\varepsilon(\lambda) - \mu_1(\lambda)$. □

Lemma 5.10. Assuming seBARG satisfies the somewhere argument of knowledge property and HT satisfies opening completeness, for every QPT attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid 1** and **Hybrid 2**.

Proof. The proof of this lemma follows from Lemma 4.8. Consequently, we can find a negligible function $\mu_2(\lambda)$ s.t the advantage of $\mathcal{A}$ in **Hybrid 2** is $\varepsilon(\lambda) - \mu_1(\lambda) - \mu_2(\lambda)$. □

Lemma 5.11. Assuming the collision resistance of the hash function used in HT with respect to the openings and hashed values, for every QPT attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid 2** and **Hybrid 3**.

Proof. The proof of this lemma follows from Lemma 4.9. Therefore, we can quantify the advantage of $\mathcal{A}$ in **Hybrid 3** using a negligible function $\mu_3(\lambda)$ as $\varepsilon(\lambda) - \mu_1(\lambda) - \mu_2(\lambda) - \mu_3(\lambda)$. □

Lemma 5.12. Assuming the correctness of FHE evaluation, for every QPT attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid 3** and **Hybrid 4**

Proof. Let the attacker $\mathcal{A}$ have non-negligible advantage across the two hybrids. This implies that

$$\mathsf{FHE.Dec}(\mathsf{fhe.sk}_{v'_0}, \mathsf{ct}^{\star}_{v'_0}) = 0$$

However, since the ATree.Build procedure produces the logical AND of the input ciphertexts, we must have $\mathsf{FHE.Dec}(\mathsf{fhe.sk}_{\epsilon}, \mathsf{ct}^{\star}) = 0$, a contradiction. Therefore, we can quantify the advantage of $\mathcal{A}$ in **Hybrid 4** using a negligible function $\mu_4(\lambda)$ as $\varepsilon(\lambda) - \mu_1(\lambda) - \mu_2(\lambda) - \mu_3(\lambda) - \mu_4(\lambda)$. □

Lemma 5.13. Assuming the correctness of FHE evaluation, for every QPT attacker $\mathcal{A}$, the attacker's advantage is the indistinguishable between **Hybrid (4,** $\ell - 1$**)** and **Hybrid (4,** ℓ**)** for all $1 \leq \ell \leq \log k$.

Proof. The follows from the proof of Lemma 5.12. Therefore, we can quantify the advantage of the attacker $\mathcal{A}$ in **Hybrid (4,** ℓ**)** using a negligible function $\nu_\ell(\lambda)$ as $\varepsilon(\lambda) - \mu_1(\lambda) - \mu_2(\lambda) - \mu_3(\lambda) - \mu_4(\lambda) - \sum_{i=0}^{\ell} \nu_i(\lambda)$ □

Lemma 5.14. Assuming the correctness of FHE evaluation, for every QPT attacker $\mathcal{A}$, the attacker's advantage is indistinguishable between **Hybrid (4,** $\log k$**)** and **Hybrid (4,** $\log k + 1$**)**

Proof. The follows from the proof of Lemma 5.12. Therefore, we can quantify the advantage of the attacker $\mathcal{A}$ in **Hybrid (4,** $\log k + 1$**)** using a negligible function $\nu_{\log k+1}(\lambda)$ as $\varepsilon(\lambda) - \mu_1(\lambda) - \mu_2(\lambda) - \mu_3(\lambda) - \mu_4(\lambda) - \sum_{i=0}^{\log k+1} \nu_i(\lambda)$. □

Lemma 5.15. Assuming the correctness of the post-quantum FHE scheme, for every QPT attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid (4,** $\log k + 1$**)** and **Hybrid 5**.

Proof. Let the attacker $\mathcal{A}$ have non-negligible advantage across the two hybrids. This implies that

$$\mathsf{VerifyInstance}(x_u, y_u^\star, z_u^\star, c, r_j) = 0 \ \wedge \ \mathsf{FHE.Dec}(\mathsf{fhe.sk}_u, \mathsf{ct}_u^\star) = 1$$

which contradicts the evaluation correctness of the FHE scheme. Therefore, the advantage of the attacker across **Hybrid (4,** $\log k + 1$**)** and **Hybrid 5** can be characterized by a negligible function $\nu_{\log k+2}(\lambda)$. Let $\mu_5(\lambda) = \sum_{i=0}^{\log k+2} \nu_i(\lambda)$. Thus, the advantage of attacker $\mathcal{A}$ in **Hybrid 5** is $\varepsilon(\lambda) - \mu_1(\lambda) - \mu_2(\lambda) - \mu_3(\lambda) - \mu_4(\lambda) - \mu_5(\lambda)$. □

Lemma 5.16. Assuming the semantic security of the post-quantum FHE scheme, for every QPT attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid 5** and **Hybrid 6**.

Proof. Assume the contrary. We show that there exists an efficient reduction algorithm $\mathcal{B}$ that breaks FHE semantic security. $\mathcal{B}$ interacts with challenger $\mathcal{C}$ for the FHE security game as follows.

- $\mathcal{B}$ receives the index $i^\star$ from the attacker $\mathcal{A}$
- $\mathcal{B}$ samples randomness $r_{i^\star}$ and sends $(r_{i^\star}, 0^\lambda)$ to $\mathcal{C}$.
- $\mathcal{C}$ replies with $\widetilde{\mathsf{ct}}$, a random encryption of one of the two messages.
- $\mathcal{B}$ replaces $\mathsf{ct}_{r_{i^\star}}$ in the crs with $\widetilde{\mathsf{ct}}$ and interacts with $\mathcal{A}$ to obtain $(\hat{z}, \pi_z, \mathsf{ct}^*, \pi_{\mathsf{ct}})$.

Observe that based on which of the two messages is encrypted in $\widetilde{\mathsf{ct}}$, we could be operating in different hybrids. Now, if $\mathcal{A}$ gets accepted, then $\mathcal{B}$ can guess $r_{i^\star}$ as the message under the encryption of $\widetilde{\mathsf{ct}}$, breaking the semantic security of the FHE scheme. Therefore, there exists a negligible function $\mu_6(\lambda)$ s.t the advantage for the attacker $\mathcal{A}$ in **Hybrid 6** must be $\varepsilon(\lambda) - \mu_1(\lambda) - \mu_2(\lambda) - \mu_3(\lambda) - \mu_4(\lambda) - \mu_5(\lambda) - \mu_6(\lambda)$. □

At this point, the attacker $\mathcal{A}$ has at least $\varepsilon(\lambda) - \mathsf{negl}(\lambda)$ chance of success, where

$$\mathsf{negl}(\lambda) = (\mu_1 + \mu_2 + \mu_3 + \mu_4 + \mu_5 + \mu_6)(\lambda)$$

Lemma 5.17. Assuming the soundness of the Commit and Measure Protocol (Maha) and the FHM Protocol, **Hybrid 6**, hence referred to as $\mathfrak{M}'$, is a Commit-Challenge-Response protocol with computationally orthogonal projectors.

Proof. The proof follows from Lemma 4.10. □

Lemma 5.18. The λ-fold repetition of $\mathfrak{M}'$, i.e $(\mathfrak{M}')^\lambda$ has negligible soundness.

Proof. This follows from Theorem 3.27. □

Lemma 5.19. The λ-fold repetition of $\mathfrak{M}$, i.e $\mathfrak{M}^\lambda$, has negligible soundness.

Proof. This proof follows the same arguments as Lemma 4.12. Therefore, we must have $\varepsilon(\lambda) = \mathsf{negl}(\lambda)$. Therefore, we note that $\mathfrak{M}^\lambda$ is a public coin BARG that satisfies the semi-adaptive soundness property. □

□

6 Making 4-round Batch Arguments for QMA fully-succinct

As outlined in the construction above, the first message of the verifier scales linearly with the number of instances. We need $2k+1$ FHE public keys, $2k$ encryption of secret keys for internal nodes of the ATree, k public keys for the Commit-and-Measure protocol, and finally k encryption of the randomness r for each instance x_i. We proceed with our construction as below.

6.1 Construction

We construct the $\widetilde{\mathsf{batS}}$ algorithm by obfuscating the $\mathsf{batchSetup}$ algorithm outlined in the construction of $\mathfrak{M}$. We rely on PPRF security and $\mathsf{i}\mathcal{O}$ security to achieve a semi-adaptive notion of soundness. We need the following additional ingredients when compared to the construction in Section 5.

Ingredients

1. Indistinguishability Obfuscation $\mathsf{i}\mathcal{O}$ as defined in Definition 3.12.
2. Two puncturable PRF families $\mathsf{PPRF}(\mathsf{Eval}, \mathsf{Puncture}, \mathsf{PuncEval})$ with output lengths $\ell(\lambda)$ and $2\ell(\lambda)$ respectively, as defined in Definition 3.9.

Batched Setup Algorithm : Let

$$\begin{aligned} F &: \{0,1\}^\lambda \times \{0,1\}^{\leq \log k} \to \{0,1\}^\lambda \\ F' &: \{0,1\}^\lambda \times \{0,1\}^{\leq \log k} \to \{0,1\}^{2\lambda} \end{aligned}$$

be two PPRF functions. We batch $\mathfrak{M}.\mathsf{Setup}$ procedure by hardwiring λ, N, k, PPRF key $K_1 \leftarrow \{0,1\}^\lambda$ associated with F and PPRF keys $K_2, K_3 \leftarrow \{0,1\}^\lambda$ associated with F' respectively. The $\mathsf{batchSetup}$ algorithm is presented in Fig. 1. Finally, to complete the succinct key generation setup, we obfuscate the $\mathsf{batchSetup}$ program, sample the crs for the seBARGs (for any random index $j^\star \in [k]$) and the hash function for the HT

$$\begin{aligned} \widetilde{\mathsf{batS}} &\leftarrow \mathsf{i}\mathcal{O}(\mathsf{batchSetup}) \\ \mathsf{ht.hk} &\leftarrow \mathsf{HT.Gen}(1^\lambda, 1^k) \\ \mathsf{sB.crs}_1 &\leftarrow \mathsf{seBARG}_1.\mathsf{Setup}(1^\lambda, 1^k, 1^{|\mathcal{C}_\mathcal{G}|}, j^\star) \\ \mathsf{sB.crs}_2 &\leftarrow \mathsf{seBARG}_2.\mathsf{Setup}(1^\lambda, 1^k, 1^{|\mathcal{C}'_\mathcal{G}|}, j^\star) \\ \mathsf{crs} &= (\widetilde{\mathsf{Setup}}, \mathsf{ht.hk}, \mathsf{sB.crs}_1, \mathsf{sB.crs}_2) \end{aligned}$$

To complete the protocol, the prover uses the program $\widetilde{\mathsf{batS}}$ to generate all the required public information for each binary index $v \in \{0,1\}^{\leq \log k}$ of the ATree. The leaf nodes of the tree represent the instances. The prover uses the computed public information and performs the rest of the protocol as in Section 5.

Remark 6.1. We denote this protocol as $\mathfrak{M}_s$ and $\mathfrak{M}_s^\lambda$ is a λ-fold repetition.

Program batchSetup

Constants: $\lambda, N, k, K_1, K_2, K_3$
Input: Index $v \in \{0,1\}^{\leq \log k}$.
Output: Public information PK_v

1. If length of v is less than $\log k$:
   (a) Let $r_v = F_{K_1}(v)$, $\{r_{v\|\beta} = F_{K_1}(v\|\beta)\}_{\beta\in\{0,1\}}$, and $(r'_{v\|0}, r'_{v\|1}) = F'_{K_2}(v)$.
   (b) Perform the following operations

$$\begin{aligned}
(\mathsf{fhe.pk}_v, \mathsf{fhe.sk}_v) &= \mathsf{FHE.KeyGen}(1^\lambda; r_v) \\
(\mathsf{fhe.pk}_{v\|\beta}, \mathsf{fhe.sk}_{v\|\beta}) &= \mathsf{FHE.KeyGen}(1^\lambda; r_{v\|\beta}),\ \beta \in \{0,1\} \\
\forall \beta \in \{0,1\}\ \mathsf{aux}_{v,\beta} &= \mathsf{FHE.Enc}(\mathsf{fhe.pk}_v, \mathsf{fhe.sk}_{v\|\beta}; r'_{v\|\beta}) \\
\mathsf{aux}_v &= (\mathsf{aux}_{v,0}, \mathsf{aux}_{v,1})
\end{aligned}$$

   (c) Output $\mathsf{PK}_v = (\mathsf{fhe.pk}_v, \mathsf{aux}_v)$

2. Else:
   (a) $r_v, r''_v = F'_{K_3}(v)$, $r'_v = F_{K_1}(v)$.
   (b) Setup the Commit-and-Measure protocol Maha as follows

$$(\mathsf{mh.pk}_v, \mathsf{mh.sk}_v) = \mathsf{Maha.Gen}(1^\lambda; r_v)$$

   (c) Sample $\mathsf{fhe.pk}_v$ and compute ct_{r_v} as follows

$$\begin{aligned}
(\mathsf{fhe.pk}_v, \mathsf{fhe.sk}_v) &= \mathsf{FHE.KeyGen}(1^\lambda; r'_v) \\
\mathsf{ct}_{r_v} &= \mathsf{FHE.Enc}(\mathsf{fhe.pk}_v, r_v; r''_v)
\end{aligned}$$

   (d) Output $\mathsf{PK}_v = (\mathsf{mh.pk}_v, \mathsf{fhe.pk}_v, \mathsf{ct}_{r_v})$

Figure 1: Batched Setup algorithm

6.2 Security Analysis

Completeness : The completeness of the protocol follows from the perfect completeness of $\mathsf{i}\mathcal{O}$ as defined in Definition 3.12.

Round Complexity : We observe that $\mathfrak{M}_s$ and therefore $\mathfrak{M}_s^\lambda$ both have **3 rounds** with setup. Moreover, if the crs is considered as the first message sent by the verifier, we observe that the message is succinct (independent of k) and we get a **4 round** Interactive **Designated Verifier** BARG for QMA

Soundness : The protocol satisfies semi-adaptive notion of soundness as defined in Definiton 3.20. We now provide the soundness proof.

Theorem 6.2. Assuming the security of $\mathsf{i}\mathcal{O}$ (Definition 3.12), Puncturable PRFs (Defintion 3.9), and the soundness of the non-succinct designated verifier BARG for QMA protocol $\mathfrak{M}$, the protocol $\mathfrak{M}_s^\lambda$ satisfies semi-adaptive soundness as defined in Definition 3.20)

Proof. We carefully organize a sequence of hybrids that modifies the Setup algorithm such that it remains functionally equivalent, but allows a challenger to sample $\log k + 1$ FHE keys needed to verify the opening from the ATree. Using the security of PPRF and $i\mathcal{O}$, we argue that any cheating prover cannot differentiate between any two successive programs. Finally, we follow the hybrids and proof of Theorem 5.8 to show a λ-fold repetition of $\mathfrak{M}_s$ must have negligible soundness. Assume a binary string based indexing scheme. Let u be the binary representation of $i^\star$ using $\log k$ bits. Let the nodes along the path from the root $\epsilon = v_0$ to u be $\{v_\ell\}$ and let the sibling nodes be $\{v'_\ell\}$ for $0 \le \ell \le \log k$. We have the following sequence of hybrids

Hybrid 0: Normal execution of the protocol.

Hybrid 1: For each $0 \le \ell \le \log k - 1$, we have four intermediary hybrids:

(a) **Hybrid (1, ℓ, 0)**: Sample $\mathsf{fhe.pk}_{v_\ell}$ as follows

$$(\mathsf{fhe.pk}_{v_\ell}, \mathsf{fhe.sk}_{v_\ell}) = \mathsf{FHE.KeyGen}(1^\lambda; F_{K_1}(v_\ell))$$

Puncture K_1 at $v = v_\ell$ and run $\mathsf{batchSetup}_0^{(\ell)}$ as in Fig. 2 by hardwiring $\mathsf{fhe.pk}_{v_\ell}$

(b) **Hybrid (1, ℓ, 1)**: Sample $\mathsf{fhe.pk}_{v_\ell}$ using fresh randomness $r \leftarrow \{0,1\}^\lambda$ as follows:

$$(\mathsf{fhe.pk}_{v_\ell}, \mathsf{fhe.sk}_{v_\ell}) = \mathsf{FHE.KeyGen}(1^\lambda; r)$$

Run $\mathsf{batchSetup}_0^{(\ell)}$ as in Fig. 2 by hardwiring $\mathsf{fhe.pk}_{v_\ell}$.

(c) **Hybrid (1, ℓ, 2)**: Perform the following operations for $v = v_\ell$ with $(r'_{v\|0}, r'_{v\|1}) = F'_{K_2}(v)$

$$\begin{aligned}(\mathsf{fhe.pk}_{v\|\beta}, \mathsf{fhe.sk}_{v\|\beta}) &= \mathsf{FHE.KeyGen}(1^\lambda; F_{K_1}(v\|\beta)),\ \beta \in \{0,1\}\\ \forall \beta \in \{0,1\},\ \mathsf{aux}_{v,v\|\beta} &= \mathsf{FHE.Enc}(\mathsf{fhe.pk}_v, \mathsf{fhe.sk}_{v\|\beta}; r'_{v\|\beta})\\ \mathsf{aux}_v &= (\mathsf{aux}_{v\|0}, \mathsf{aux}_{v\|1})\end{aligned}$$

Puncture K_2 at $v = v_\ell$ and run $\mathsf{batchSetup}_1^{(\ell)}$ as in Fig. 3 by hardwiring aux_v.

(d) **Hybrid (1, ℓ, 3)**: Perform the following operations for $v = v_\ell$ using fresh randomness $r'_{v\|0}, r'_{v\|1} \leftarrow \{0,1\}^\lambda$:

$$\begin{aligned}(\mathsf{fhe.pk}_{v\|\beta}, \mathsf{fhe.sk}_{v\|\beta}) &= \mathsf{FHE.KeyGen}(1^\lambda; F_{K_1}(v\|\beta)),\ \beta \in \{0,1\}\\ \forall \beta \in \{0,1\},\ \mathsf{aux}_{v,v\|\beta} &= \mathsf{FHE.Enc}(\mathsf{fhe.pk}_v, \mathsf{fhe.sk}_{v\|\beta}; r'_{v\|\beta})\\ \mathsf{aux}_v &= (\mathsf{aux}_{v\|0}, \mathsf{aux}_{v\|1})\end{aligned}$$

Run $\mathsf{batchSetup}_1^{(\ell)}$ as in Fig. 3 by hardwiring aux_v.

Hybrid 2: We have two intermediate hybrids:

(a) **Hybrid (2, 0)**: Sample $\mathsf{fhe.pk}_u$ as follows

$$(\mathsf{fhe.pk}_u, \mathsf{fhe.sk}_u) = \mathsf{FHE.KeyGen}(1^\lambda; F_{K_1}(u))$$

Puncture K_1 at $v = u$ and run $\mathsf{batchSetup}_2^{(\log k)}$ as in Fig. 4 by hardwiring $\mathsf{fhe.pk}_u$

(b) **Hybrid (2, 1)**: Sample $\mathsf{fhe.pk}_u$ using fresh randomness $r \leftarrow \{0,1\}^\lambda$ as follows

$$(\mathsf{fhe.pk}_u, \mathsf{fhe.sk}_u) = \mathsf{FHE.KeyGen}(1^\lambda; r)$$

Run $\mathsf{batchSetup}_2^{(\log k)}$ as in Fig. 4 by hardwiring $\mathsf{fhe.pk}_u$.

Hybrid 3: We have two intermediate hybrids:

(a) **Hybrid (3, 0)**: Perform the following operations $r_u, r''_u = F'_{K_3}(u)$

$$(\mathsf{mh.pk}_u, \mathsf{mh.sk}_u) = \mathsf{Maha.Gen}(1^\lambda; r_u)$$
$$\mathsf{ct}_{r_u} = \mathsf{FHE.Enc}(\mathsf{fhe.pk}_u, r_u; r''_u)$$

Puncture K_3 at $v = u$ and run $\mathsf{batchSetup}_3^{(\log k)}$ as in Fig. 5 by hardwiring $(\mathsf{mh.pk}_u, \mathsf{ct}_{r_u})$.

(b) **Hybrid (3,1)**: Perform the following operations using fresh randomness $r, r'' \leftarrow \{0,1\}^\lambda$

$$(\mathsf{mh.pk}_u, \mathsf{mh.sk}_u) = \mathsf{Maha.Gen}(1^\lambda; r)$$
$$\mathsf{ct}_{r_u} = \mathsf{FHE.Enc}(\mathsf{fhe.pk}_u, r; r'')$$

Run $\mathsf{batchSetup}_3^{(\log k)}$ as in Fig. 5 by hardwiring $(\mathsf{mh.pk}_u, \mathsf{ct}_{r_u})$.

Hybrid 4-9: Follows **Hybrid 1-6** in Theorem 5.8.

We now prove that the attacker's advantage negligibly changes through the sequence of hybrids assuming the soundness properties of the underlying building blocks. Subsequently, we show, like in Theorem 5.8 that **Hybrid 9** is a Commit-and-Measure protocol with orthogonal projects. Thus any parallel repetition of **Hybrid 9** has negligible soundness, thus showing that $\mathfrak{M}_s^\lambda$ has negligible soundness. For contradiction, assume that a cheating prover has non-negligible advantage $\varepsilon(\lambda)$ in breaking **Hybrid 0**. Consider the following lemmas.

Lemma 6.3. Assuming the security of Indistinguishability Obfuscation $\mathsf{i}\mathcal{O}$, for any poly-size attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid 0** and **Hybrid (1,0,0)**

Proof. We observe that the input/output behavior of $\mathsf{batchSetup}_0^{(0)}$ is identical to that of $\mathsf{batchSetup}$. We only puncture the PPRF key K_1 at $v = \epsilon$ and replace it with the hardwired value $\mathsf{fhe.pk}_\epsilon$. For contradiction, assume $\mathcal{A}$ has non-negligible advantage across **Hybrid 0** and **Hybrid (1,0,0)**. Consider a reduction algorithm $\mathcal{B}$ that receives the obfuscated program $\widetilde{\mathsf{batS}}$ from the challenger $\mathcal{C}$ where $\mathsf{batchSetup}$ or $\mathsf{batchSetup}_0^0$ is chosen uniformly at random. Since $\mathcal{A}$ has non-negligible advantage across the two hybrids, $\mathcal{B}$ uses $\mathcal{A}$ trivially to guess which program was obfuscated, contradicting the security of $\mathsf{i}\mathcal{O}$. Therefore, there exists a negligible function $\nu_{0,0}(\lambda)$ s.t the advantage of the attacker $\mathcal{A}$ in **Hybrid (1,0,0)** is $\varepsilon(\lambda) - \nu_{0,0}(\lambda)$. □

Lemma 6.4. Assuming the security of PPRF, for any poly-size attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid (1, ℓ, 0)** and **Hybrid (1, ℓ, 1)** for all $0 \leq \ell \leq \log k - 2$

Proof. The only change between **Hybrid (1, ℓ, 0)** and **Hybrid (1, ℓ, 1)** is the randomness used to generate $\mathsf{fhe.pk}_\epsilon$. For contradiction, assume that $\mathcal{A}$ has non-negligible advantage across the two hybrids. Construct a reduction algorithm $\mathcal{B}$ that interacts trivially with $\mathcal{A}$ to produce distributions $(F_{K_1\{\epsilon,\ldots,v_\ell\}}(v), r)$ and successfully differentiate between the two, contradicting the security of PPRFs. Therefore, there exists a negligible function $\nu_{\ell,1}(\lambda)$ s.t the advantage of the attacker $\mathcal{A}$ in **Hybrid (1, ℓ, 1)** is $\varepsilon(\lambda) - (\sum_{k=0}^{\ell-1} \nu_k(\lambda) + \eta_k(\lambda)) - \nu_{\ell,0}(\lambda) - \nu_{\ell,1}(\lambda)$. □

Program $\mathsf{batchSetup}_0^{(\ell)}$

Constants: $\lambda, N, k, K_1\{\epsilon, v_1, \ldots, v_{\ell-1}, v_\ell\}, K_2\{\epsilon, v_1, \ldots, v_{\ell-1}\}, K_3$
Hardwired values: $\{\mathsf{fhe.pk}_\epsilon, \ldots, \mathsf{fhe.pk}_{v_{\ell-1}}, \mathsf{fhe.pk}_{v_\ell}\}$, $\{\mathsf{aux}_\epsilon, \ldots, \mathsf{aux}_{v_{\ell-1}}\}$
Input: Index $v \in \{0,1\}^{\leq \log k}$.
Output: Public information PK_v

1. If length of v is less than $\log k$:
   (a) If $v \in \{\epsilon, \ldots, v_{\ell-1}\}$, output $\mathsf{PK}_v = (\mathsf{fhe.pk}_v, \mathsf{aux}_v)$.
   (b) Else if $v = v_\ell$:
      i. Let $\{r_{v\|\beta} = F_{K_1}(v\|\beta)\}_{\beta\in\{0,1\}}$. Let $(r'_{v\|0}, r'_{v\|1}) = F'_{K_2}(v)$.
      ii. Perform the following operations:

$$\begin{aligned}(\mathsf{fhe.pk}_{v\|\beta}, \mathsf{fhe.sk}_{v\|\beta}) &= \mathsf{FHE.KeyGen}(1^\lambda; r_{v\|\beta}),\ \beta \in \{0,1\}\\ \forall \beta \in \{0,1\},\ \mathsf{aux}_{v,\beta} &= \mathsf{FHE.Enc}(\mathsf{fhe.pk}_v, \mathsf{fhe.sk}_{v\|\beta}; r'_{v\|\beta})\\ \mathsf{aux}_v &= (\mathsf{aux}_{v,0}, \mathsf{aux}_{v,1})\end{aligned}$$

      iii. Output $\mathsf{PK}_v = (\mathsf{fhe.pk}_v, \mathsf{aux}_v)$.
   (c) Else:
      i. Let $r_v = F_{K_1}(v)$, $\{r_{v\|\beta} = F_{K_1}(v\|\beta)\}_{\beta\in\{0,1\}}$. Let $(r'_{v\|0}, r'_{v\|1}) = F'_{K_2}(v)$.
      ii. Perform the following operations:

$$\begin{aligned}(\mathsf{fhe.pk}_v, \mathsf{fhe.sk}_v) &= \mathsf{FHE.KeyGen}(1^\lambda; r_v)\\ (\mathsf{fhe.pk}_{v\|\beta}, \mathsf{fhe.sk}_{v\|\beta}) &= \mathsf{FHE.KeyGen}(1^\lambda; r_{v\|\beta}),\ \beta \in \{0,1\}\\ \forall \beta \in \{0,1\},\ \mathsf{aux}_{v,\beta} &= \mathsf{FHE.Enc}(\mathsf{fhe.pk}_v, \mathsf{fhe.sk}_{v\|\beta}; r'_{v\|\beta})\\ \mathsf{aux}_v &= (\mathsf{aux}_{v,0}, \mathsf{aux}_{v,1})\end{aligned}$$

      iii. Output $\mathsf{PK}_v = (\mathsf{fhe.pk}_v, \mathsf{aux}_v)$
2. Else:
   (a) $r_v, r''_v = F'_{K_3}(v)$, $r'_v = F_{K_1}(v)$.
   (b) Setup $(\mathsf{mh.pk}_v, \mathsf{mh.sk}_v) = \mathsf{Maha.Gen}(1^\lambda; r_v)$
   (c) Sample $\mathsf{fhe.pk}_v$ and compute ct_{r_v} as follows

$$\begin{aligned}(\mathsf{fhe.pk}_v, \mathsf{fhe.sk}_v) &= \mathsf{FHE.KeyGen}(1^\lambda; r'_v)\\ \mathsf{ct}_{r_v} &= \mathsf{FHE.Enc}(\mathsf{fhe.pk}_v, r_v; r''_v)\end{aligned}$$

   (d) Output $\mathsf{PK}_v = (\mathsf{mh.pk}_v, \mathsf{fhe.pk}_v, \mathsf{ct}_{r_v})$

Figure 2: **Program** $\mathsf{batchSetup}_0^{(\ell)}$: Hardwire the $\mathsf{fhe.pk}_{v_\ell}$

Lemma 6.5. Assuming the security of Indistinguishability Obfuscation $\mathsf{i}\mathcal{O}$, for any poly-size attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid (1, ℓ, 1)** and **Hybrid (1, ℓ, 2)** for all $0 \leq \ell \leq \log k - 2$

Proof. By the same reasoning as in Lemma 6.3 for key K_2 and programs $\mathsf{batchSetup}_0^{(\ell)}$ and $\mathsf{batchSetup}_1^{(\ell)}$, there exists a negligible function $\nu_{\ell,2}(\lambda)$ s.t the advantage of the attacker $\mathcal{A}$ in **Hybrid (1, ℓ, 2)** is $\varepsilon(\lambda) - (\sum_{k=0}^{\ell-1} \nu_k(\lambda) + \eta_k(\lambda)) - \nu_{\ell,0}(\lambda) - \nu_{\ell,1}(\lambda) - \nu_{\ell,2}(\lambda)$. □

Lemma 6.6. Assuming the security of PPRF, for any poly-size attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid (1, ℓ, 2)** and **Hybrid (1, ℓ, 3)** for all $0 \leq \ell \leq \log k - 2$

Proof. By the same reasoning as Lemma 6.4 for key K_2, there exists a negligible function $\nu_{\ell,3}(\lambda)$

Program $\mathsf{batchSetup}_1^{(\ell)}$

Constants: $\lambda, N, k, K_1\{\epsilon, v_1, \ldots, v_\ell\}, K_2\{\epsilon, v_1, \ldots, v_{\ell-1}, v_\ell\}, K_3$
Hardwired values: $\{\mathsf{fhe.pk}_\epsilon, \ldots, \mathsf{fhe.pk}_{v_\ell}\}$, $\{\mathsf{aux}_\epsilon, \ldots, \mathsf{aux}_{v_{\ell-1}}, \mathsf{aux}_{v_\ell}\}$
Input: Index $v \in \{0,1\}^{\leq \log k}$.
Output: Public information PK_v

1. If length of v is less than $\log k$:
   (a) If $v \in \{\epsilon, v_1, \ldots, v_{\ell-1}\}$, output $\mathsf{PK}_v = (\mathsf{fhe.pk}_v, \mathsf{aux}_v)$.
   (b) Else if $v = v_\ell$, output $\mathsf{PK}_v = (\mathsf{fhe.pk}_v, \mathsf{aux}_v)$.
   (c) Else:
      i. Let $r_v = F_{K_1}(v)$, $\{r_{v\|\beta} = F_{K_1}(v\|\beta)\}_{\beta\in\{0,1\}}$. Let $(r'_{v\|0}, r'_{v\|1}) = F'_{K_2}(v)$.
      ii. Perform the following operations:

$$\begin{aligned}(\mathsf{fhe.pk}_v, \mathsf{fhe.sk}_v) &= \mathsf{FHE.KeyGen}(1^\lambda; r_v)\\ (\mathsf{fhe.pk}_{v\|\beta}, \mathsf{fhe.sk}_{v\|\beta}) &= \mathsf{FHE.KeyGen}(1^\lambda; r_{v\|\beta}),\ \beta \in \{0,1\}\\ \forall \beta \in \{0,1\},\ \mathsf{aux}_{v,\beta} &= \mathsf{FHE.Enc}(\mathsf{fhe.pk}_v, \mathsf{fhe.sk}_{v\|\beta}; r'_{v\|\beta})\\ \mathsf{aux}_v &= (\mathsf{aux}_{v,0}, \mathsf{aux}_{v,1})\end{aligned}$$

      iii. Output $\mathsf{PK}_v = (\mathsf{fhe.pk}_v, \mathsf{aux}_v)$
2. Else:
   (a) $r_v, r''_v = F'_{K_3}(v)$, $r'_v = F_{K_1}(v)$.
   (b) Setup $(\mathsf{mh.pk}_v, \mathsf{mh.sk}_v) = \mathsf{Maha.Gen}(1^\lambda; r_v)$
   (c) Sample $\mathsf{fhe.pk}_v$ and compute ct_{r_v} as follows

$$\begin{aligned}(\mathsf{fhe.pk}_v, \mathsf{fhe.sk}_v) &= \mathsf{FHE.KeyGen}(1^\lambda; r'_v)\\ \mathsf{ct}_{r_v} &= \mathsf{FHE.Enc}(\mathsf{fhe.pk}_v, r_v; r''_v)\end{aligned}$$

   (d) Output $\mathsf{PK}_v = (\mathsf{mh.pk}_v, \mathsf{fhe.pk}_v, \mathsf{ct}_{r_v})$

Figure 3: **Program** $\mathsf{batchSetup}_1^{(\ell)}$: Hardwire aux_{v_ℓ}

s.t the advantage of the attacker $\mathcal{A}$ in **Hybrid (1, ℓ, 3)** is $\varepsilon(\lambda) - (\sum_{k=0}^{\ell-1} \nu_k(\lambda) + \eta_k(\lambda)) - \nu_{\ell,0}(\lambda) - \nu_{\ell,1}(\lambda) - \nu_{\ell,2}(\lambda) - \nu_{\ell,3}(\lambda) = \varepsilon(\lambda) - (\sum_{k=0}^{\ell-1} \nu_k(\lambda) + \eta_k(\lambda)) - \nu_\ell(\lambda)$. □

Lemma 6.7. Assuming the security of Indistinguishability Obfuscation $i\mathcal{O}$, for any poly-size attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid (1, ℓ, 3)** and **Hybrid (1, $\ell+1$, 0)** for all $0 \leq \ell \leq \log k - 2$

Proof. By the same reasoning as Lemma 6.3 for key K_1, there exists a negligible function $\eta_\ell(\lambda)$ s.t the advantage of the attacker $\mathcal{A}$ in **Hybrid (1, $\ell+1$, 0)** is $\varepsilon(\lambda) - (\sum_{k=0}^{\ell-1} \nu_k(\lambda) + \eta_k(\lambda)) - \eta_\ell(\lambda) = \varepsilon(\lambda) - (\sum_{k=0}^{\ell} \nu_k(\lambda) + \eta_k(\lambda))$. □

Lemma 6.8. Assuming the security of Indistinguishability Obfuscation $i\mathcal{O}$ and the security of PPRF, for any poly-size attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid (1, $\log k - 1$, 0)** and **Hybrid (1, $\log k - 1$, 3)**.

Proof. We follow the sequence of hybrids **Hybrid (1, $\log k - 1$, 0)** to **Hybrid (1, $\log k - 1$, 3)** using the same reasoning as in Lemma 6.3- Lemma 6.7. Therefore, there exists a negligible function $\nu_{\log k-1}(\lambda)$ s.t the advantage of the attacker $\mathcal{A}$ in **Hybrid (1, $\log k - 1$, 3)** is $\varepsilon(\lambda) - (\sum_{k=0}^{\log k-2} \nu_k(\lambda) + \eta_k(\lambda)) - \nu_{\log k-1}(\lambda) = \varepsilon(\lambda) - \mu_1(\lambda)$ □

Program $\mathsf{batchSetup}_2^{(\log k)}$

Constants: $\lambda, N, k, K_1\{\epsilon, v_1, \ldots, v_{\log k-1}, u\}, K_2\{\epsilon, v_1, \ldots, v_{\log k-1}\}, K_3$
Hardwired values: $\{\mathsf{fhe.pk}_\epsilon, \ldots, \mathsf{fhe.pk}_{v_{\log k}}, \mathsf{fhe.pk}_u\}$, $\{\mathsf{aux}_\epsilon, \ldots, \mathsf{aux}_{v_{\log k-1}}\}$
Input: Index $v \in \{0,1\}^{\leq \log k}$.
Output: Public information PK_v

1. If length of v is less than $\log k$:
   (a) If $v \in \{\epsilon, v_1, \ldots, v_{\log k-1}\}$, output $\mathsf{PK}_v = (\mathsf{fhe.pk}_v, \mathsf{aux}_v)$.
   (b) Else:
      i. Let $r_v = F_{K_1}(v)$, $\{r_{v\|\beta} = F_{K_1}(v\|\beta)\}_{\beta\in\{0,1\}}$. Let $(r'_{v\|0}, r'_{v\|1}) = F'_{K_2}(v)$.
      ii. Perform the following operations:

$$\begin{aligned}(\mathsf{fhe.pk}_v, \mathsf{fhe.sk}_v) &= \mathsf{FHE.KeyGen}(1^\lambda; r_v)\\ (\mathsf{fhe.pk}_{v\|\beta}, \mathsf{fhe.sk}_{v\|\beta}) &= \mathsf{FHE.KeyGen}(1^\lambda; r_{v\|\beta}),\ \beta \in \{0,1\}\\ \forall \beta \in \{0,1\},\ \mathsf{aux}_{v,\beta} &= \mathsf{FHE.Enc}(\mathsf{fhe.pk}_v, \mathsf{fhe.sk}_{v\|\beta}; r'_{v\|\beta})\\ \mathsf{aux}_v &= (\mathsf{aux}_{v,0}, \mathsf{aux}_{v,1})\end{aligned}$$

      iii. Output $\mathsf{PK}_v = (\mathsf{fhe.pk}_v, \mathsf{aux}_v)$
2. Else:
   (a) Let $r_v, r''_v = F'_{K_3}(v)$.
   (b) Setup $(\mathsf{mh.pk}_v, \mathsf{mh.sk}_v) = \mathsf{Maha.Gen}(1^\lambda; r_v)$
   (c) If $v = u$:
      i. Compute $\mathsf{ct}_{r_u} = \mathsf{FHE.Enc}(\mathsf{fhe.pk}_u, r_v; r''_v)$.
      ii. Output $\mathsf{PK}_u = (\mathsf{mh.pk}_u, \mathsf{fhe.pk}_u, \mathsf{ct}_{r_u})$
   (d) Else:
      i. Let $r'_v = F_{K_1}(v)$.
      ii. Sample $\mathsf{fhe.pk}_v$ and compute ct_{r_v} as follows

$$\begin{aligned}(\mathsf{fhe.pk}_v, \mathsf{fhe.sk}_v) &= \mathsf{FHE.KeyGen}(1^\lambda; r'_v)\\ \mathsf{ct}_{r_v} &= \mathsf{FHE.Enc}(\mathsf{fhe.pk}_v, r_v; r''_v)\end{aligned}$$

      iii. Output $\mathsf{PK}_v = (\mathsf{mh.pk}_v, \mathsf{fhe.pk}_v, \mathsf{ct}_{r_v})$

Figure 4: **Program** $\mathsf{batchSetup}_2^{(\log k)}$: Hardwire $\mathsf{fhe.pk}_u$

Lemma 6.9. Assuming the security of Indistinguishability Obfuscation $i\mathcal{O}$, for any poly-size attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid (1, $\log k - 1$, 3)** and **Hybrid (2, 0)**.

Proof. By the same reasoning as in Lemma 6.3 for key K_1 and programs $\mathsf{batchSetup}_1^{(\log k)}$ and $\mathsf{batchSetup}_2^{(\log k)}$, there exists a negligible function $\nu_{2,0}(\lambda)$ s.t the advantage of the attacker $\mathcal{A}$ in **Hybrid (2, 0)** is $\varepsilon(\lambda) - \mu_1(\lambda) - \nu_{2,0}(\lambda)$. □

Lemma 6.10. Assuming the security of PPRF, for any poly-size attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid (2, 0)** and **Hybrid (2, 1)**

Proof. By the same reasoning as Lemma 6.4 for key K_1, there exists a negligible function $\nu_{2,1}(\lambda)$ s.t advantage of attacker in **Hybrid (2, 1)** is $\varepsilon(\lambda) - \mu_1(\lambda) - \nu_{2,0}(\lambda) - \nu_{2,1}(\lambda) = \varepsilon(\lambda) - \mu_1(\lambda) - \mu_2(\lambda)$. □

Program $\mathsf{batchSetup}_3^{(\log k)}$

Constants: $\lambda, N, k, K_1\{\epsilon, v_1, \ldots, v_{\log k-1}, u\}, K_2\{\epsilon, v_1, \ldots, v_{\log k-1}\}, K_3\{u\}$
Hardwired values: $\{\mathsf{fhe.pk}_\epsilon, \ldots, \mathsf{fhe.pk}_{v_{\log k-1}}, \mathsf{fhe.pk}_u\}$, $\{\mathsf{aux}_\epsilon, \ldots, \mathsf{aux}_{v_{\log k-1}}\}, \mathsf{mh.pk}_u, \mathsf{ct}_{r_u}$
Input: Index $v \in \{0,1\}^{\leq \log k}$.
Output: Public information PK_v

1. If length of v is less than $\log k$:
   (a) If $v \in \{\epsilon, v_1, \ldots, v_{\log k-1}\}$, output $\mathsf{PK}_v = (\mathsf{fhe.pk}_v, \mathsf{aux}_v)$.
   (b) Else:
      i. Let $r_v = F_{K_1}(v)$, $\{r_{v\|\beta} = F_{K_1}(v\|\beta)\}_{\beta\in\{0,1\}}$. Let $(r'_{v\|0}, r'_{v\|1}) = F'_{K_2}(v)$.
      ii. Perform the following operations:

$$\begin{aligned}(\mathsf{fhe.pk}_v, \mathsf{fhe.sk}_v) &= \mathsf{FHE.KeyGen}(1^\lambda; r_v)\\ (\mathsf{fhe.pk}_{v\|\beta}, \mathsf{fhe.sk}_{v\|\beta}) &= \mathsf{FHE.KeyGen}(1^\lambda; r_{v\|\beta}),\ \beta \in \{0,1\}\\ \forall \beta \in \{0,1\},\ \mathsf{aux}_{v,\beta} &= \mathsf{FHE.Enc}(\mathsf{fhe.pk}_v, \mathsf{fhe.sk}_{v\|\beta}; r'_{v\|\beta})\\ \mathsf{aux}_v &= (\mathsf{aux}_{v,0}, \mathsf{aux}_{v,1})\end{aligned}$$

      iii. Output $\mathsf{PK}_v = (\mathsf{fhe.pk}_v, \mathsf{aux}_v)$
2. Else:
   (a) If $v = u$, output $(\mathsf{mh.pk}_u, \mathsf{fhe.pk}_u, \mathsf{ct}_{r_u})$
   (b) Else:
      i. Let $r_v, r''_v = F'_{K_3}(v)$. Let $r'_v = F_{K_1}(v)$.
      ii. Setup $(\mathsf{mh.pk}_v, \mathsf{mh.sk}_v) = \mathsf{Maha.Gen}(1^\lambda; r_v)$
      iii. Sample $\mathsf{fhe.pk}_v$ and compute ct_{r_v} as follows

$$\begin{aligned}(\mathsf{fhe.pk}_v, \mathsf{fhe.sk}_v) &= \mathsf{FHE.KeyGen}(1^\lambda; r'_v)\\ \mathsf{ct}_{r_v} &= \mathsf{FHE.Enc}(\mathsf{fhe.pk}_v, r_v; r''_v)\end{aligned}$$

      iv. Output $\mathsf{PK}_v = (\mathsf{mh.pk}_v, \mathsf{fhe.pk}_v, \mathsf{ct}_{r_v})$

Figure 5: **Program** $\mathsf{batchSetup}_3^{(\log k)}$: Hardwire $(\mathsf{mh.pk}_u, \mathsf{ct}_{r_u})$

Lemma 6.11. Assuming the security of Indistinguishability Obfuscation $\mathsf{i}\mathcal{O}$, for any poly-size attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid (2, 1)** and **Hybrid (3, 0)**.

Proof. By the same reasoning as in Lemma 6.3 for key K_1 and programs $\mathsf{batchSetup}_2^{(\log k)}$ and $\mathsf{batchSetup}_3^{(\log k)}$, there exists a negligible function $\nu_{3,0}(\lambda)$ s.t the advantage of the attacker $\mathcal{A}$ in **Hybrid (3, 0)** is $\varepsilon(\lambda) - \mu_1(\lambda) - \mu_2(\lambda) - \nu_{3,0}(\lambda)$. □

Lemma 6.12. Assuming the security of PPRF, for any poly-size attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid (3, 0)** and **Hybrid (3, 1)**

Proof. By the same reasoning as Lemma 6.4 for key K_3, there exists a negligible function $\nu_{3,1}(\lambda)$ s.t the advantage of the attacker in **Hybrid (3, 1)** is $\varepsilon(\lambda) - \mu_1(\lambda) - \mu_2(\lambda) - \nu_{3,0}(\lambda) - \nu_{3,1}(\lambda) = \varepsilon(\lambda) - \mu_1(\lambda) - \mu_2(\lambda) - \mu_3(\lambda)$. □

Lemma 6.13. Assuming the semantic security and evaluation correctness of FHE, collision resistance of HT, index hiding and somewhere argument of knowledge property of seBARGs, the attacker's advantage is indistinguishable between all hybrids from **Hybrid 4** to **Hybrid 9**.

Proof. Follows from Lemma 5.9 to Lemma 5.16. □

Lemma 6.14. Assuming the soundness of Commit-and-Measure protocol and the FHM protocol, **Hybrid 9** is a Commit challenge response protocol with computationally orthogonal projectors

Proof. Follows from Lemma 5.17. □

Lemma 6.15. The λ-fold repetition of $\mathfrak{M}_s$, i.e, $\mathfrak{M}_s^\lambda$ has negligible soundness.

Proof. Follows from Lemma 5.18 and Lemma 5.19. Therefore, we must have $\varepsilon(\lambda) = \mathsf{negl}(\lambda)$ and $\mathfrak{M}_s^\lambda$ is a public coin protocol (except for the first message) BARG that satisfies the semi-adaptive soundness property. □

□

7 Monotone Policy Batch Arguments for QMA

We first begin by defining a Monotone Policy Batch QMA language as an analogue to the definition presented [BBK+23, Section. 4]. Consider any Monotone Policy Batch language $\mathcal{L}_f^{(k)}$ (corresponding **quantum** relation $\mathcal{R}$) defined using a function f with a Monotone Circuit Family $\mathfrak{C}$ as follows

$$\mathcal{L}_f^{(k)} = \left\{ (x_1, \ldots, x_k) : \exists(|\phi_1\rangle, \ldots, |\phi_k\rangle) \text{ s.t } f(b_1, \ldots, b_k) = 1, \text{ where } b_i = \mathcal{R}(x_i, |\phi_i\rangle) \right\}$$

Let $C_f \in \mathfrak{C}$ be the circuit computing f. We impose the same restrictions on C_f as imposed by [BBK+23, Section. 4]. Crucially, we bound the width and size of the circuit to $\mathsf{poly}(\lambda)$. We now, propose the following theorem.

Theorem 7.1 (Interactive Monotone Policy Batch Arguments for QMA)**.** Assuming the post-quantum hardness of the learning with errors problem (LWE), there is a **6-round** private-coin interactive argument system for any monotone policy batch of k T-time quantum computations on inputs $x_1, \ldots, x_k$ over a function f (corresponding monotone circuit C_f) and corresponding $N = \mathsf{poly}(T)$ qubit witnesses $|\phi_1\rangle, \ldots, |\phi_k\rangle$, where

1. The Prover P is a QPT algorithm that runs in time $\mathsf{poly}(k, N)$
2. The Verifier V is a PPT algorithm that runs in time $\mathsf{poly}(\log k, N)$
3. The length of each message in the protocol is bounded by $\mathsf{poly}(\log k, N)$

The Interactive argument satisfies the completeness and a **Non-Adaptive** notion of soundness as defined below.

Definition 7.2 (**Monotone** BARG **Completeness**)**.** For an interactive argument (P, V) with $k = k(\lambda)$, $N = N(\lambda)$, and $x = (x_1, x_2, \ldots, x_k) \in \mathcal{L}_f^{(k)}$, and any corresponding quantum witnesses $|\phi\rangle = (|\phi_1\rangle, |\phi_2\rangle, \ldots, |\phi_k\rangle)$, there exists a negligible function $\mathsf{negl}(\cdot)$ s.t it holds for every $\lambda \in \mathbb{N}$

$$\Pr\left[\mathsf{Out}\left\langle \mathsf{P}(x, |\phi\rangle), \mathsf{V}(x)\right\rangle = 1\right] \leq 1 - \mathsf{negl}(\lambda)$$

Definition 7.3 (**Non-Adaptive Soundness**)**.** For any $k = k(\lambda)$, $N = N(\lambda)$, for any QPT attacker $\mathsf{P}^\star$, for any function f with a monotone circuit C_f, any set of instances $x = (x_1, x_2, \ldots, x_k) \notin \mathcal{L}_f^{(k)}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that it holds for every $\lambda \in \mathbb{N}$,

$$\Pr\left[\mathsf{Out}\left\langle \mathsf{P}^\star(x), \mathsf{V}(x)\right\rangle = 1\right] \leq \mathsf{negl}(\lambda)$$

7.1 Construction

Ingredients

1. Predicate Extractable Hash Family with Tags for bit-fixing predicates $\mathsf{PEHT}(\mathsf{Gen}, \mathsf{Hash}, \mathsf{Open}, \mathsf{Verify}, \mathsf{Extract}, \mathsf{ExtractIndex}, \mathsf{ExtractTag})$ as in Definition 3.3. We modify the $\mathsf{ExtractIndex}$ function to output the lexicographically smallest index in J, i.e we define the following procedure:

   $\mathsf{ExtractIndex}(\mathsf{td}, \mathsf{v})$:

   (a) Parse $\mathsf{td} = J, \mathsf{sk}$.

   (b) Output $(b, j, t) = \mathsf{FHE.Dec}(\mathsf{sk}, \mathsf{v})$

   (c) Output j, or the lexicographically smallest index of J if $j \notin J$.

2. Predicate Extractable Hash Family $\mathsf{PEHash}(\mathsf{Gen}, \mathsf{Hash}, \mathsf{Open}, \mathsf{Verify}, \mathsf{Extract}, \mathsf{ExtractIndex})$ for the restricted set of bit-fixing predicates with succinct description. Let $\mathcal{F}$ be a family of bit-fixing predicates s.t there exists a poly-time TM $\mathcal{M}_{\mathcal{F}}$ that receives an input a circuit C_f, description d_f of a predicate $f \in \mathcal{F}$ and outputs the necessary set and an assignment (J, m). The TM $\mathcal{M}_{\mathcal{F}}$ performs the following operations:

   (a) Parse $d_f = (x, i)$, where $x \in \{0,1\}^k$ is an input to the circuit $i \in [d]$ is a layer in the circuit C_f.

   (b) Let $(b_1^\star, b_2^\star, \ldots, b_M^\star)$ be the values on all wires of C_f. Here, $M = |C_f|$.

   (c) Let $J \subseteq M$ be the set of wires in the ith layer of C_f whose values $b^\star$ are 0, and let $m \in \{0,1\}^J$ be the all-zero vector

   (d) Output (J, m)

3. Somewhere Extractable BARG or $\mathsf{seBARG}(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Extract})$. We use of an instance of seBARG where Setup supports a constant number of indices instead of just one.
4. Commit-and-Measure Protocol $\mathsf{Maha}(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open}, \mathsf{Test}, \mathsf{Out})$ as defined in Definition 3.21.
5. The κ-fold FHM post hoc verification protocol $(\mathsf{P}_{\mathsf{FHM}}, \mathsf{V}_{\mathsf{FHM}})$ as defined in Definition 3.30 where κ is a polynomial in instance length.

To simplify the presentation, we make use of the following languages.

Definition 7.4 (Consistency Language)**.** A consistency language $\mathcal{N}_{\mathsf{vk}_1,\mathsf{vk}_2}$ is defined as follows

$$\mathcal{N}_{\mathsf{vk}_1,\mathsf{vk}_2} = \big\{(i, \mathsf{v}_1, \mathsf{v}_2) \mid \exists\ b, \alpha, \beta \text{ such that } \mathsf{PEHT.Verify}(\mathsf{vk}_1, \mathsf{v}_1, i, b, \alpha) = 1 \wedge \mathsf{PEHT.Verify}(\mathsf{vk}_2, \mathsf{v}_2, i, b, \beta) = 1\big\}$$

Note that we have overloaded the $\mathsf{PEHT.Verify}$ function to not need the tags. With this language, we want to ensure that the string used to compute v_1 and v_2 is the same. Let $\mathcal{R}_{\mathcal{N}}$ denote a relation for $\mathcal{N}_{\mathsf{vk}_1,\mathsf{vk}_2}$ s.t $\mathcal{R}_{\mathcal{N}}(i, \mathsf{v}_1, \mathsf{v}_2) = 1$ iff $(i, \mathsf{v}_1, \mathsf{v}_2) \in \mathcal{N}_{\mathsf{vk}_1,\mathsf{vk}_2}$. Let the corresponding circuit encoding this relation be $\mathcal{C}_{\mathcal{N}}$.

Definition 7.5 (Batch Index Satisfiability TM χ)**.** We define a TM $\chi(z, j, \omega_j)$ for an input $z = (C, x, \mathsf{pT.vk}, \mathsf{pT.v}, \mathsf{pH.vk}, \mathsf{pH.v}, \mathsf{h}, c, r)$, to perform the following computations:

1. Parse $z = (C, x, \mathsf{pT.vk}, \mathsf{pT.v}, \mathsf{pH.vk}, \mathsf{pH.v}, \mathsf{h}, c, r)$ and $x = (x_1, x_2, \ldots, x_k)$.

2. Parse the following

$$\begin{aligned}\mathsf{pT.vk} &= (\mathsf{pT.vk}_1, \mathsf{pT.vk}_2)\\ \mathsf{pH.vk} &= (\mathsf{pH.vk}_1, \mathsf{pH.vk}_2)\\ \mathsf{pH.v} &= (\mathsf{pH.v}_1, \mathsf{pH.v}_2)\\ \mathsf{h} &= (\mathsf{h}_1, \mathsf{h}_2)\end{aligned}$$

3. If $1 \leq j \leq k$:
   (a) Parse $\omega_j = (w_j, b_j, \mathsf{op}_j, \rho_j^{(1)}, \rho_j^{(2)})$ where $w_j = (\mathsf{op}_j^{(1)}, \mathsf{op}_j^{(2)}, t_j^{(1)}, t_j^{(2)}, b'_j)$.
   (b) Check if $\mathsf{PEHT.Verify}(\mathsf{pT.vk}_2, \mathsf{pT.v}, j, b_j, w_j, \mathsf{op}_j) = 1$
   (c) Check if $\mathsf{PEHash.Verify}(\mathsf{pH.vk}_\beta, \mathsf{pH.v}_\beta, j, b_j, \rho_j^{(\beta)}, C) = 1$ for $\beta \in [2]$.
   (d) Check if $\mathsf{PEHT.Verify}(\mathsf{pT.vk}_1, \mathsf{h}_\beta, j, b'_j, \mathsf{op}_j^{(\beta)}) = 1$ for $\beta \in [2]$
   (e) If $b'_j = 1$, check if $\mathsf{VerifyInstance}(x_j, t_j^{(1)}, t_j^{(2)}, c, r) = 1$
4. If $k < j \leq M$:
   (a) Compute the jth gate of C, $G_j = (j, j_1, j_2, g \in \{\text{AND}, \text{OR}\})$.
   (b) Parse $\omega_j = \left(b_j, b_{j_1}, b_{j_2}, \left(\rho_j^{(\beta)}, \rho_{j_1}^{(\beta)}, \rho_{j_2}^{(\beta)}\right)_{\beta \in [2]}\right)$.
   (c) Check if $\mathsf{PEHash.Verify}(\mathsf{pH.vk}_\beta, \mathsf{pH.v}_\beta, j, b_j, \rho_j^{(\beta)}, C) = 1$ for $\beta \in [2]$.
   (d) Check if $\mathsf{PEHash.Verify}(\mathsf{pH.vk}_\beta, \mathsf{pH.v}_\beta, j_\alpha, b_{j_\alpha}, \rho_{j_\alpha}^{(\beta)}, C) = 1$ for $\alpha, \beta \in [2]$
   (e) Check if $b_j = g(b_{j_1}, b_{j_2})$
5. If $j = M$ is the output wire, check that $b_j = 1$

Clearly, the description of χ is constant. Compute $\tau > \mathsf{poly}(M, k)$ large enough that the pseudocode above terminates.

Definition 7.6 (Monotone Circuit Evaluation Language)**.** A Monotone Circuit Evaluation language $\mathcal{K}_\chi$ is the set of all instances that is accepted by the Turing machine χ in time τ.

Protocol Specification

- $\mathsf{Setup}(1^\lambda, 1^k, 1^N, f) \to (\mathsf{crs}, \mathsf{vk})$: Let C_f be the monotone circuit for f.
  - Setup the commit-and-measure protocol Maha. Sample randomness $r \xleftarrow{\$} \{0,1\}^\lambda$ and subsequently sample the basis string h.

$$(\mathsf{mh.pk}, \mathsf{mh.sk}) = \mathsf{Maha.Gen}(1^\lambda, h; r)$$

– For $i \in [2]$, setup PEHT_i as follows for any random subset $J \subset [k]$ and $m = 0^J$

$$\begin{aligned}(\mathsf{pT.hk}_i, \mathsf{pT.vk}_i, \mathsf{pT.td}_i) &\leftarrow \mathsf{PEHT}_i.\mathsf{Gen}(1^\lambda, k, J, m)\\ \mathsf{pT.hk} &= (\mathsf{pT.hk}_1, \mathsf{pT.hk}_2)\\ \mathsf{pT.vk} &= (\mathsf{pT.vk}_1, \mathsf{pT.vk}_2)\\ \mathsf{pT.td} &= (\mathsf{pT.td}_1, \mathsf{pT.td}_2)\end{aligned}$$

We have the tag length for $(\mathsf{pT.hk}_1, \mathsf{pT.vk}_1, \mathsf{pT.td}_1)$ being set to N, the number of qubits in a single copy of the witness state. The tag length for $(\mathsf{pT.hk}_2, \mathsf{pT.vk}_2, \mathsf{pT.td}_2)$ is bounded by a number $T = \mathcal{O}(N \cdot \mathsf{poly}(\lambda))$.

– Let $M = |C_f|$ and d_{f_i} is the description of an arbitrary predicate $f_i \in \mathcal{F}$. For $i \in [2]$, generate

$$\begin{aligned}(\mathsf{pH.hk}_i, \mathsf{pH.vk}_i, \mathsf{pH.td}_i) &\leftarrow \mathsf{PEHash.Gen}(1^\lambda, M, d_{f_i})\\ \mathsf{pH.hk} &= (\mathsf{pH.hk}_1, \mathsf{pH.hk}_2)\\ \mathsf{pH.vk} &= (\mathsf{pH.vk}_1, \mathsf{pH.vk}_2)\\ \mathsf{pH.td} &= (\mathsf{pH.td}_1, \mathsf{pH.td}_2)\end{aligned}$$

– Generate $\mathsf{sB.crs}_1$ for seBARG_1 with any index $j \in J$

$$(\mathsf{sB.crs}_1, \mathsf{sB.td}_1) \leftarrow \mathsf{seBARG}_1.\mathsf{Setup}(1^\lambda, 1^k, 1^{|\mathcal{C}_\mathcal{N}|}, j)$$

Generate $\mathsf{sB.crs}_2$ for seBARG_2 for the Monotone circuit evaluation language $\mathcal{K}_\chi$ and set $I = \{1, 2, N\}$

$$(\mathsf{sB.crs}_2, \mathsf{sB.td}_2) \leftarrow \mathsf{seBARG}_2.\mathsf{Setup}(1^\lambda, 1^M, 1^T, I)$$

Compute

$$\mathsf{sB.crs} = (\mathsf{sB.crs}_1, \mathsf{sB.crs}_2), \qquad \mathsf{sB.td} = (\mathsf{sB.td}_1, \mathsf{sB.td}_2)$$

– Outputs the following.

$$\begin{aligned}\mathsf{crs} &= (\mathsf{mh.pk}, \mathsf{pT.hk}, \mathsf{pT.vk}, \mathsf{pH.hk}, \mathsf{pH.vk}, \mathsf{sB.crs})\\ \mathsf{vk} &= (\mathsf{mh}, \mathsf{sk}, r)\end{aligned}$$

- $\mathsf{Prove}(\mathsf{crs}, \{x_j\}_{j\in[k]}, \{|\phi_j\rangle^{\otimes\kappa}\}_{j\in[k]})$: Here, the quantum state $|\phi_i\rangle$ is the quantum witness to the statement x_i for some $\mathcal{L} \in \mathsf{QMA}$. The prover computes the string s s.t $s_i = (x_i \in \mathcal{L}_{\text{yes}})$ s.t $f(s) = 1$ and proceeds as follows:

  – **Round 1**: Parse crs to get $(\mathsf{pT.hk}_1, \mathsf{mh.pk})$. If $s_j = 0$, set $y_j = \perp$, else compute

$$\begin{aligned}|\psi_j\rangle &\leftarrow \mathsf{P}_{\mathsf{FHM}}(x_j, |\phi_j\rangle^{\otimes\kappa})\\ (y_j, |\mathsf{st}_j\rangle) &\leftarrow \mathsf{Maha.Com}(\mathsf{mh.pk}, |\psi_j\rangle)\end{aligned}$$

Finally calculate h_y as follows and send to the verifier.

$$\mathsf{h}_y = \mathsf{PEHT}_1.\mathsf{Hash}(\mathsf{pT.hk}_1, s, \{y_j\}_{j\in[k]})$$

– **Round 3**: Receive a bit $c \in \{0,1\}$ from the verifier. Parse crs to get $\mathsf{sB.crs}_1$. Let

$$z_j = \begin{cases} \bot & s_j = 0, \\ \mathsf{Maha.Open}(|\mathsf{st}_j\rangle, c) & s_j = 1 \end{cases}$$

Compute h_z as follows:

$$\mathsf{h}_z = \mathsf{PEHT}_1.\mathsf{Hash}(\mathsf{pT.hk}_1, s, \{z_j\}_{j\in[k]})$$

Generate the seBARG_1 proof and send (h_z, π_1) to the verifier.

$$\begin{aligned} \forall j,\ \mathsf{op}_j^y &= \mathsf{PEHT}_1.\mathsf{Open}(\mathsf{pT.hk}_1, s, \{y_j\}_{j\in[k]}, j) \\ \forall j,\ \mathsf{op}_j^z &= \mathsf{PEHT}_1.\mathsf{Open}(\mathsf{pT.hk}_1, s, \{z_j\}_{j\in[k]}, j) \\ \forall j,\ v_j &= (\mathsf{op}_j^y, \mathsf{op}_j^z, y_j, z_j, s_j) \\ \pi_1 &\leftarrow \mathsf{seBARG}_1.\mathsf{Prove}(\mathsf{sB.crs}_1, \{(j, \mathsf{h}_y, \mathsf{h}_z)\}_{j\in[k]}, \{v_j\}_{j\in[k]}) \end{aligned}$$

– **Round 5**: Receive $r \in \{0,1\}^\lambda$ from the verifier. Perform the following

1. Compute the values of all the wires of circuit C_f. Let these values be $b_1, \ldots, b_M$ with the first k bits is set to s_i, i.e $b_i = s_i$ for all $i \in [k]$.
2. Parse crs to get $(\mathsf{pT.hk}_2, \mathsf{pT.vk}_2, \mathsf{pH.hk}_1, \mathsf{pH.hk}_2, \mathsf{pH.vk}_1, \mathsf{pH.vk}_2, \mathsf{sB.crs}_2)$
3. Compute the following

$$\begin{aligned} \{w_j\}_{j\in[k]} &= \{(\mathsf{op}_j^y, y_j, \mathsf{op}_j^z, z_j, s_j)\}_{j\in[k]} \\ \mathsf{h}_w &= \mathsf{PEHT}_2.\mathsf{Hash}(\mathsf{pT.hk}_2, s, \{w_j\}_{j\in[k]}) \end{aligned}$$

4. Compute the following

$$\begin{aligned} \mathsf{v}_i &= \mathsf{PEHash.Hash}(\mathsf{pH.hk}_i, (b_1, b_2, \ldots, b_M), C_f), \qquad i \in [2] \\ \mathsf{pH.v} &= (\mathsf{v}_1, \mathsf{v}_2) \end{aligned}$$

5. For $1 \le j \le k$, perform the following tasks

$$\begin{aligned} \mathsf{op}_j^w &= \mathsf{PEHT}_2.\mathsf{Open}(\mathsf{pT.hk}_2, s, \{w_j\}_{j\in[k]}, j) \\ \rho_j^{(i)} &= \mathsf{PEHash.Open}(\mathsf{pH.hk}_i, b_j, j, C_f), \quad i \in [2] \end{aligned}$$

For $k < j \le M$, perform the following tasks for gate $G_j = (j, j_1, j_2, g)$

$$\begin{aligned} \rho_j^{(i)} &= \mathsf{PEHash.Open}(\mathsf{pH.hk}_i, b_j, j, C_f), \quad i \in [2] \\ \rho_{j_\alpha}^{(i)} &= \mathsf{PEHash.Open}(\mathsf{pH.hk}_i, b_{j_\alpha}, j_\alpha, C_f), \quad \alpha, i \in [2] \end{aligned}$$

Construct the witness ω_j for all $j \in [M]$ as follows

$$\omega_j = \begin{cases} (w_j, s_j, \mathsf{op}_j^w, \rho_j^{(1)}, \rho_j^{(2)}) & 1 \le j \le k \\ \left(b_j, b_{j_1}, b_{j_2}, \left(\rho_j^{(\beta)}, \rho_{j_1}^{(\beta)}, \rho_{j_2}^{(\beta)}\right)_{\beta\in[2]}\right) & k < j \le M \end{cases}$$

6. Compute the seBARG_2 proof as follows

$$\begin{aligned} x &= (x_1, x_2, \ldots, x_k) \\ \mathsf{h} &= (\mathsf{h}_y, \mathsf{h}_z) \\ \zeta &= (C_f, x, \mathsf{pT.vk}, \mathsf{h}_w, \mathsf{pH.vk}, \mathsf{pH.v}, \mathsf{h}, c, r) \\ \pi_2 &\leftarrow \mathsf{seBARG}_2.\mathsf{Prove}(\mathsf{sB.crs}_2, \{(j,\zeta)\}_{j\in[M]}, \{\omega_j\}_{j\in[M]}) \end{aligned}$$

7. Compute the consistency proof. Send $(\mathsf{h}_w, \mathsf{v}_1, \mathsf{v}_2, \pi_2, \pi_3)$ to verifier

$$\begin{aligned} \{u_j\}_{j\in[k]} &= \{(\mathsf{op}_j^w, \mathsf{op}_j^z, w_j, z_j, s_j)\}_{j\in[k]} \\ \pi_3 &\leftarrow \mathsf{seBARG}_1(\mathsf{sB.crs}_1, \{(j, \mathsf{h}_w, \mathsf{h}_z)\}_{j\in[k]}, \{u_j\}_{j\in[k]}) \end{aligned}$$

- $\mathsf{Verify}(\mathsf{crs}, \mathsf{vk}, \{x_j\}_{j\in[k]}, \mathsf{h}_y, \mathsf{h}_z, \mathsf{h}_w, \mathsf{v}_1, \mathsf{v}_2, \pi_1, \pi_2, \pi_3)$: Works as follows under normal (honest) execution.
  - **Round 2**: Receive h_y from prover. Sample bit c and send it to the prover.
  - **Round 4**: Receive h_z from prover. Parse crs to get $\mathsf{sB.crs}_1$ and check if

$$\mathsf{seBARG}_1.\mathsf{Verify}(\mathsf{sB.crs}_1, \{(j, \mathsf{h}_y, \mathsf{h}_z)\}, \pi_1) = 1$$

    If yes, parse vk and send r to the prover. Else, reject.
  - **Final Verification**: Receive $(\mathsf{h}_w, \mathsf{v}_1, \mathsf{v}_2, \pi_2, \pi_3)$ from the prover. Parse crs to get ($\mathsf{sB.crs}_2$, $\mathsf{pT.vk}_2, \mathsf{pH.vk}_1, \mathsf{pH.vk}_2$). Construct ζ as presented previously. Accept if

$$\begin{aligned} \mathsf{seBARG}_1.\mathsf{Verify}(\mathsf{sB.crs}_1, \{(j, \mathsf{h}_z, \mathsf{h}_w)\}_{j\in[k]}, \pi_3) = 1 \\ \mathsf{seBARG}_2.\mathsf{Verify}(\mathsf{sB.crs}_2, \{(j,\zeta)\}_{j\in[M]}, \pi_2) = 1 \end{aligned}$$

    Otherwise, reject.

Remark 7.7. We denote the above protocol as $\mathfrak{R}$ and $\mathfrak{R}^\lambda$ is a λ-fold repetition.

7.2 Security Analysis

Completeness For any monotone policy f for the instances $x = (x_1, \ldots, x_k)$ the interaction results in the verifier accepting with non-negligible probability if $x \in \mathcal{L}_f^{(k)}$ by the completeness properties of the underlying primitives. The completeness of $\mathfrak{R}^\lambda$ follows by a union-bound argument.

Round Complexity We observe that $\mathfrak{R}$ and (therefore) $\mathfrak{R}^\lambda$ both have **5 rounds** with setup. Alternatively, the verifier can do the setup and crs can be sent to the prover as the verifier's first message to get a **6 round Private Coin** Interactive Monotone Policy BARG for QMA.

Before we proceed to prove soundness, we define the following property for PEHTs.

Definition 7.8. A Predicate Extractable Hash Family with Tags w.r.t a bit-fixing predicate class $\mathcal{F}$ satisfies the following property

Lexicographically Smallest Index Extraction: For any poly-sized adversary $\mathcal{A}$, for any polynomial $N = N(\lambda)$ and any bit fixing predicate described by a set $J \subset [N]$ and string $y \in \{0,1\}^J$, for any two indices $i_1, i_2 \in J$ s.t $i_1 < i_2$, there exists a negligible function $\mathsf{negl}(\cdot)$ s.t for any $\lambda \in \mathbb{N}$,

$$\Pr\left[\begin{array}{l} \mathsf{Extract}(\mathsf{td},\mathsf{v}) = 0 \wedge \\ \mathsf{ExtractIndex}(\mathsf{td},\mathsf{v}) = i_2 \wedge \\ \mathsf{Verify}(\mathsf{vk},\mathsf{v},i_1,1-y_{i_1},t,\rho) = 1 \end{array} : \begin{array}{l} (\mathsf{hk},\mathsf{vk},\mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda,N,J,y) \\ (\mathsf{v},i_1,t,\rho) \leftarrow \mathcal{A}(\mathsf{hk},\mathsf{vk}) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Lemma 7.9. Assuming the correctness of evaluation of FHE, the function $\mathsf{PEHT.ExtractIndex}(\mathsf{td},\mathsf{v})$ outputs the lexicographically smallest index $i \in J$ as defined in Definition 7.8.

Proof. We refer the reader to the construction of the PEHT [BBK+23, Section. 5]. By the modification introduced in our construction, when $\mathsf{PEHT.Extract}(\mathsf{td},\mathsf{v}) = 1$, we always return the lexicographically smallest index. We are left to show that we do the same when $\mathsf{PEHT.Extract}(\mathsf{td},\mathsf{v}) = 0$. Now, for contradiction, assume that the output of ExtractIndex is an index $i_2 \in J$ that is not the lexicographically smallest index (say the index is i_1, i.e., $i_1 < i_2$). Let v be the longest common prefix in the binary representations of i_1 and i_2. At node v, both children $v\|0$ and $v\|1$ must return $(0, i_1, t_{i_1})$ and $(0, i_2, t_{i_2})$, respectively. To compute the ciphertext at v, the function evaluated under FHE must choose the minimum index since both $i_1, i_2 \in J$. If ExtractIndex outputs i_2, we contradict the correctness of FHE. □

Soundness The protocol $\mathfrak{R}^\lambda$ satisfies the notion of **non-adaptive soundness** as defined in Definition 7.3. We now provide the soundness proof.

Theorem 7.10. Assuming the soundness of the Commit-and-Measure protocol (Definition 3.23) and the FHM protocol (Definition 3.32), computational binding and extraction correctness of tag and index of PEHT, PEHash, and somewhere argument of knowledge property of seBARG, the protocol $\mathfrak{R}^\lambda$ satisfies *non-adaptive computational soundness* as defined in Definition 7.3.

Proof. The proof strategy for the construction proceeds as follows. First, we note that $\mathfrak{R}$ is a *Commit-Challenge-Response* protocol as per Definition 3.24. For non-adaptive soundness, the challenger $\mathcal{C}$ and attacker $\mathcal{A}$ are given the instance $x = (x_1, x_2, \ldots, x_k) \notin \mathcal{L}_f^{(k)}$ and the monotone circuit C_f pair (x, C_f). Additionally, the challenger $\mathcal{C}$ also receives the necessary set $J^\star \subset [k]$ for C_f as non-uniform advice. The necessary set $J^\star$ contains a subset of indices in $[k]$ of which one (say i) must satisfy $\mathcal{R}(x_i, |\phi_i\rangle) = 1$ for $C_f(x) = 1$. We say that the attacker $\mathcal{A}$ wins in the entire security game if the challenger $\mathcal{C}$ accepts for the pair (x, C_f) even though $C_f(x) = 0$.

We set up a sequence of hybrids to show the consistency between the hash values in **Round 1** and **Round 3**. Then, we show that the extractions in **Round 5** are valid via a proof that follows the one presented in [BBK+23]. We argue further that the hashes calculated in **Round 3** and **Round 5** are consistent with each other. Finally, we show that the final hybrid has a negligible advantage over the original protocol. We proceed as follows:

Hybrid 0: Normal execution of the protocol

Hybrid 1: Same as previous execution except that we now enable extraction on the necessary set $J^\star$ for PEHT_1, i.e

$$(\mathsf{pT.hk}_1^\star, \mathsf{pT.vk}_1^\star, \mathsf{pT.td}_1^\star) \leftarrow \mathsf{PEHT}_1.\mathsf{Gen}(1^\lambda, k, J^\star, 0^{J^\star})$$

Hybrid 2: Same as previous execution, except we add the following check in **Round 4** of the protocol

$$\mathsf{PEHT}_1.\mathsf{Extract}(\mathsf{pT.td}_1^\star, \mathsf{h}_y) = 0 \ \wedge \ \mathsf{PEHT}_1.\mathsf{Extract}(\mathsf{pT.td}_1^\star, \mathsf{h}_z) = 0$$

Hybrid 3: We now update the protocol by extracting the tags and indices in **Round 4** on the verifier's end. Reject if $j_1, j_2 \notin J^\star$

$$\begin{aligned} j_1 = \mathsf{PEHT}_1.\mathsf{ExtractIndex}(\mathsf{pT.td}_1^\star, \mathsf{h}_y) \quad y_{j_1}^\star = \mathsf{PEHT}_1.\mathsf{ExtractTag}(\mathsf{pT.td}_1^\star, \mathsf{h}_y) \\ j_2 = \mathsf{PEHT}_1.\mathsf{ExtractIndex}(\mathsf{pT.td}_1^\star, \mathsf{h}_z) \quad z_{j_2}^\star = \mathsf{PEHT}_1.\mathsf{ExtractTag}(\mathsf{pT.td}_1^\star, \mathsf{h}_z) \end{aligned}$$

Hybrid 4: We reject if $j_1 \neq j_2$ in **Round 4**.

Hybrid 5: Same as previous execution except that we now enable extraction on the necessary set $J^\star$ for PEHT_2, i.e.

$$(\mathsf{pT.hk}_2^\star, \mathsf{pT.vk}_2^\star, \mathsf{pT.td}_2^\star) \leftarrow \mathsf{PEHT}_2.\mathsf{Gen}(1^\lambda, k, J^\star, 0^{J})$$

Hybrid 6: Same as previous execution except we add the following check in the final verification round.

$$\mathsf{PEHT}_2.\mathsf{Extract}(\mathsf{pT.td}_2^\star, \mathsf{h}_w) = 0$$

Hybrid 7: We update the protocol by extracting the tag and index in the final verification round. Reject if $j_3 \notin J^\star$

$$\begin{aligned} j_3 = \mathsf{PEHT}_2.\mathsf{ExtractIndex}(\mathsf{pT.td}_2^\star, \mathsf{h}_w) \\ w_{j_3} = (\mathsf{op}_{j_3}^{y\star}, \mathsf{op}_{j_3}^{z\star}, y_{j_3}^{\star\prime}, z_{j_3}^{\star\prime}, s_{j_3}^\star) = \mathsf{PEHT}_2.\mathsf{ExtractTag}(\mathsf{pT.td}_2^\star, \mathsf{h}_w) \end{aligned}$$

Hybrid 8: We update the protocol to reject if any of the checks are false.

$$\begin{aligned} \mathsf{PEHT}_1.\mathsf{Verify}(\mathsf{pT.td}_1^\star, \mathsf{h}_y, j_3, s_{j_3}^\star, \mathsf{op}_{j_3}^{y\star}) &= 1 \ \wedge \\ \mathsf{PEHT}_1.\mathsf{Verify}(\mathsf{pT.td}_1^\star, \mathsf{h}_z, j_3, s_{j_3}^\star, \mathsf{op}_{j_3}^{z\star}) &= 1 \ \wedge \\ \mathsf{VerifyInstance}(x_{j_3}, y_{j_3}^{\star\prime}, z_{j_3}^{\star\prime}, c, r) = 1 \ \wedge \ s_{j_3}^\star &= 1 \end{aligned}$$

Hybrid 9: We reject if $j_2 \neq j_3$ and $j_1 \neq j_2$ in the final verification round.

We show that **Hybrid 9** is a Commit-Challenge-Response protocol that has orthogonal projectors. We then show that the attacker's advantage has changed negligibly from **Hybrid 0** to **Hybrid 9**, ensuring that $\mathfrak{R}^\lambda$ cannot have non-negligible soundness. To begin, let us assume that the attacker's advantage at **Hybrid 0** is $\varepsilon(\lambda)$. We wish to show that $\varepsilon(\lambda)$ is a negligible function.

Lemma 7.11. Assuming the predicate hiding property of PEHT_1, for any QPT attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid 0** and **Hybrid 1**.

Proof. Let us assume that the attacker $\mathcal{A}$ in **Hybrid 1** has a non-negligible advantage over an attacker in **Hybrid 0**. We construct a reduction algorithm $\mathcal{B}$ to break the predicate-hiding property of PEHT using $\mathcal{A}$. Let $\mathcal{C}$ be the challenger for the predicate hiding property for PEHT_1. We perform the following operations

$$\begin{aligned} J &\leftarrow \mathcal{B}(1^k) \\ (\mathsf{pT.hk}'_1, \mathsf{pT.vk}'_1) &\leftarrow \mathcal{C}(J, J^\star) \\ (\mathsf{h}'_y, \mathsf{h}'_z) &\leftarrow \mathcal{A}(\mathsf{pT.hk}'_1, \mathsf{pT.vk}'_1) \end{aligned}$$

If the challenger accepts in the final round, then $\mathcal{B}$ guesses that $J^\star$ was used to generate ($\mathsf{pT.hk}'_1$, $\mathsf{pT.vk}'_1$) rather than J contradicting the predicate hiding property of PEHT_1. Therefore, there exists a negligible function $\mu_1(\lambda)$ s.t the advantage of $\mathcal{A}$ in **Hybrid 1** is $\varepsilon(\lambda) - \mu_1(\lambda)$ □

Lemma 7.12. Assuming extraction correctness of PEHT_1, for any QPT attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid 1** and **Hybrid 2**

Proof. Let us assume that the attacker in **Hybrid 2** has non-negligible advantage over the attacker in **Hybrid 1**. Since $x \notin \mathcal{L}_f^{(k)}$ and $C_f(x) = 0$, we can allow a reduction algorithm $\mathcal{B}$ to interact trivially with the attacker in **Hybrid 2** to break the extraction correctness property of PEHT. Therefore, there exists a negligible function $\mu_2(\lambda)$ s.t the advantage of $\mathcal{A}$ in **Hybrid 2** is $\varepsilon(\lambda) - \mu_1(\lambda) - \mu_2(\lambda)$ □

Lemma 7.13. Assuming index extraction correctness of PEHT_1, for any QPT attacker $\mathcal{A}$, the attacker's advantage doesn't change in **Hybrid 2** and **Hybrid 3**

Proof. Any attacker that can output indices $j_1, j_2 \notin J$ must break the index extraction correctness property of PEHT. As the probability of this event is 0, the advantage of the attacker $\mathcal{A}$ in **Hybrid 3** is $\varepsilon(\lambda) - \mu_1(\lambda) - \mu_2(\lambda)$ □

Lemma 7.14. Assuming index hiding property and somewhere argument of knowledge properties for seBARG_1, consistency of extraction, and consistency of tag extraction for PEHT_1, for any QPT attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid 3** and **Hybrid 4**.

Proof. For contradiction, assume that the advantage of the attacker between **Hybrid 3** and **Hybrid 4** is $\mu_3(\lambda)$, a non-negligible function. Let the advantage in the individual hybrids be $\delta_3(\lambda)$ and $\delta_4(\lambda)$ respectively, s.t $\mu_3(\lambda) = \delta_4(\lambda) - \delta_3(\lambda)$. We say that $\mathcal{A}$ wins if the attacker can output two indices $j_1, j_2 \in [k]$ in **Hybrid 3** s.t $j_1 \neq j_2$ in **Hybrid 4**. We define a series of experiments that are performed by the challenger $\mathcal{C}$ in both the hybrids. We denote each experiment as $\mathsf{Expt}_{x,0}$ denoting that the experiment is performed on **Hybrid x**. They are as follows

1. $\mathsf{Expt}_{x,0}$: Update **Hybrid x** by guessing an index $i^\star \in [k]$ s.t $i^\star = \min(j_1, j_2)$. Abort otherwise.
2. $\mathsf{Expt}_{x,1}$: Update the previous experiment and enable extraction for seBARG_1 on index $i^\star$, i.e

$$(\mathsf{sB.crs}_1^\star, \mathsf{sB.td}_1^\star) \leftarrow \mathsf{seBARG}_1.\mathsf{Setup}(1^\lambda, 1^k, 1^{|\mathcal{C}_\mathcal{N}|}, i^\star)$$

3. $\mathsf{Expt}_{x,2}$: Update the previous experiment to extract the witness for the seBARG proof π_1

$$(\mathsf{op}_{i^\star}^{y\prime}, \mathsf{op}_{i^\star}^{z\prime}, y'_{i^\star}, z'_{i^\star}, s'_{i^\star}) \leftarrow \mathsf{seBARG.Extract}(\mathsf{sB.td}_1^\star, \{(j, \mathsf{h}_y, \mathsf{h}_z)\}_{j\in[k]}, \pi_1)$$

Without loss of generality, let $i^\star = j_1$, i.e we assume that $j_1 < j_2$. We verify the validity of the witness at the end of **Round 4**. Abort otherwise

$$\mathsf{PEHT}_1.\mathsf{Verify}(\mathsf{pT.vk}_1^\star, \mathsf{h}_y, i^\star, s'_{i^\star}, \mathsf{op}_{i^\star}^{y\prime}) = 1$$
$$\mathsf{PEHT}_1.\mathsf{Verify}(\mathsf{pT.vk}_1^\star, \mathsf{h}_z, i^\star, s'_{i^\star}, \mathsf{op}_{i^\star}^{z\prime}) = 1$$

4. $\mathsf{Expt}_{x,3}$: Update the previous experiment by aborting if $s'_{i^\star} = 0$.

Claim 7.15. The advantage of the attacker $\mathcal{A}$ between experiments $\mathsf{Expt}_{3,0}$ and $\mathsf{Expt}_{4,0}$ is given by

$$\left|\mathsf{Pr}\left[\mathsf{Expt}_{4,0}^{\mathcal{A}}\right] - \mathsf{Pr}\left[\mathsf{Expt}_{3,0}^{\mathcal{A}}\right]\right| \geq \frac{\mu_3(\lambda)}{|J^\star|}$$

Proof. This follows from a simple probability argument since $i^\star$ is essentially chosen uniformly at random from the set $J^\star$ and $|J^\star| \leq k$. □

Claim 7.16. Assuming index hiding property of seBARGs, for any QPT attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in $\mathsf{Expt}_{4,1}$ and $\mathsf{Expt}_{3,1}$, i.e there is a negligible function $\nu_1(\lambda)$ s.t

$$\left|\mathsf{Pr}\left[\mathsf{Expt}_{4,1}^{\mathcal{A}}\right] - \mathsf{Pr}\left[\mathsf{Expt}_{3,1}^{\mathcal{A}}\right]\right| \geq \frac{\mu_3(\lambda)}{|J^\star|} - \nu_1(\lambda)$$

Proof. For any negligible function $\nu_{1,0}(\lambda)$, we have the following

$$\left|\mathsf{Pr}\left[\mathsf{Expt}_{3,1}^{\mathcal{A}}\right] - \mathsf{Pr}\left[\mathsf{Expt}_{3,0}^{\mathcal{A}}\right]\right| \geq \frac{\delta_3(\lambda)}{|J^\star|} - \nu_{1,0}(\lambda)$$

following from the proof of Lemma 4.7. Similarly, we can show that there exists a negligible function $\nu_{1,1}(\lambda)$ s.t

$$\left|\mathsf{Pr}\left[\mathsf{Expt}_{4,1}^{\mathcal{A}}\right] - \mathsf{Pr}\left[\mathsf{Expt}_{4,0}^{\mathcal{A}}\right]\right| \geq \frac{\delta_4(\lambda)}{|J^\star|} - \nu_{1,1}(\lambda)$$

The claim follows from a simple triangle inequality with $\nu_1(\lambda) = \nu_{1,1}(\lambda) - \nu_{1,0}(\lambda)$. □

Claim 7.17. Assuming the somewhere argument of knowledge property of seBARG_1, for any QPT attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in $\mathsf{Expt}_{4,2}$ and $\mathsf{Expt}_{3,2}$, i.e there is a negligible function $\nu_2(\lambda)$ s.t

$$\left|\mathsf{Pr}\left[\mathsf{Expt}_{4,2}^{\mathcal{A}}\right] - \mathsf{Pr}\left[\mathsf{Expt}_{3,2}^{\mathcal{A}}\right]\right| \geq \frac{\mu_3(\lambda)}{|J^\star|} - \nu_1(\lambda) - \nu_2(\lambda)$$

Proof. Follows similarly from the proof of Lemma 4.8 and Claim 7.16. □

Claim 7.18. Assuming the consistency of extraction property of PEHT_1, for any QPT attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in $\mathsf{Expt}_{4,3}$ and $\mathsf{Expt}_{3,3}$, i.e there is a negligible function $\nu_3(\lambda)$ s.t

$$\left|\mathsf{Pr}\left[\mathsf{Expt}_{4,3}^{\mathcal{A}}\right] - \mathsf{Pr}\left[\mathsf{Expt}_{3,3}^{\mathcal{A}}\right]\right| \geq \frac{\mu_3(\lambda)}{|J^\star|} - \nu_1(\lambda) - \nu_2(\lambda) - \nu_3(\lambda)$$

Proof. We first consider a reduction algorithm $\mathcal{B}$ that interacts with a challenger $\mathcal{C}$ for $\mathsf{Expt}_{3,3}$. $\mathcal{B}$ produces an opening $(\mathsf{h}_y, i^\star, y'_{i^\star}, \mathsf{op}_{i^\star}^{y'})$ by interacting trivially with $\mathcal{A}$ s.t we have

$$\Pr\left[\begin{array}{l}\mathsf{PEHT}_1.\mathsf{Extract}(\mathsf{pT.td}_1^\star, \mathsf{h}_y) = 0 \wedge \\ \mathsf{PEHT}_1.\mathsf{ExtractIndex}(\mathsf{pT.td}_1^\star, \mathsf{h}_y) = i^\star \wedge \\ \mathsf{PEHT}_1.\mathsf{Verify}(\mathsf{pT.vk}_1^\star, \mathsf{h}_y, i^\star, s'_{i^\star}, y'_{i^\star}, \mathsf{op}_{i^\star}^{y'}) = 1\end{array}\right] \geq \frac{\delta_3(\lambda)}{|J^\star|} - \nu_1(\lambda) - \nu_2(\lambda)$$

contradicting the consistency of extraction property of PEHT_1. Therefore, there exists a negligible function $\nu_{3,0}(\lambda)$ s.t

$$\left|\Pr\left[\mathsf{Expt}_{3,3}^{\mathcal{A}}\right] - \Pr\left[\mathsf{Expt}_{3,2}^{\mathcal{A}}\right]\right| \geq \frac{\delta_3(\lambda)}{|J^\star|} - \nu_{1,0}(\lambda) - \nu_{2,0}(\lambda) - \nu_{3,0}(\lambda)$$

Similarly, we have

$$\left|\Pr\left[\mathsf{Expt}_{4,3}^{\mathcal{A}}\right] - \Pr\left[\mathsf{Expt}_{4,2}^{\mathcal{A}}\right]\right| \geq \frac{\delta_4(\lambda)}{|J^\star|} - \nu_{1,1}(\lambda) - \nu_{2,1}(\lambda) - \nu_{3,1}(\lambda)$$

The claim follows by a simple triangle inequality with $\nu_3(\lambda) = \nu_{3,1}(\lambda) - \nu_{3,0}(\lambda)$. □

Claim 7.19. $\mu_3(\lambda)$ must be a negligible function.

Proof. Follows from Lemma 7.9, since $s'_{i^\star}$ is the same across both h_y and h_z and $i^\star, j_2 \in J$. □

Therefore, the advantage of attacker $\mathcal{A}$ in **Hybrid 4** is $\varepsilon(\lambda) - \mu_1(\lambda) - \mu_2(\lambda) - \mu_3(\lambda)$. □

Lemma 7.20. Assuming the predicate hiding property of PEHT_2, for any QPT attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid 4** and **Hybrid 5**.

Proof. Follows from the proof of Lemma 7.11. Therefore, there exists a negligible function $\mu_4(\lambda)$ s.t the advantage of attacker $\mathcal{A}$ in **Hybrid 5** is $\varepsilon(\lambda) - \mu_1(\lambda) - \mu_2(\lambda) - \mu_3(\lambda) - \mu_4(\lambda)$. □

Lemma 7.21. Assuming extraction correctness of PEHT_2, for any QPT attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid 5** and **Hybrid 6**

Proof. Follows from the proof of Lemma 7.12. Therefore, there exists a negligible function $\mu_5(\lambda)$ s.t the advantage of attacker $\mathcal{A}$ in **Hybrid 6** is $\varepsilon(\lambda) - \mu_1(\lambda) - \mu_2(\lambda) - \mu_3(\lambda) - \mu_4(\lambda) - \mu_5(\lambda)$. □

Lemma 7.22. Assuming index extraction correctness of PEHT_1, for any QPT attacker $\mathcal{A}$, the attacker's advantage doesn't change in **Hybrid 6** and **Hybrid 7**

Proof. Follows from Lemma 7.13. The advantage of the attacker $\mathcal{A}$ in **Hybrid 7** is $\varepsilon(\lambda) - \mu_1(\lambda) - \mu_2(\lambda) - \mu_3(\lambda) - \mu_4(\lambda) - \mu_5(\lambda)$. □

Lemma 7.23. Assuming the index hiding and extraction correctness properties of seBARG_2, extraction consistency, and consistency of tag extraction properties of PEHT_2 and the predicate hiding property of PEHash, for any QPT attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid 7** and **Hybrid 8**

Proof. The proof for the above lemma amounts to proving somewhere argument of knowledge for Monotone Policy Batch SNARGs where the witness is the extraction output of $\mathsf{PEHT}_2.\mathsf{ExtractTag}$. We provide the proof in Appendix B (see also [BBK+23, Theorem. 7.1]). Therefore, there exists a negligible function $\mu_6(\lambda)$ s.t, the advantage of the attacker $\mathcal{A}$ in **Hybrid 8** is $\varepsilon(\lambda) - \mu_1(\lambda) - \mu_2(\lambda) - \mu_3(\lambda) - \mu_4(\lambda) - \mu_5(\lambda) - \mu_6(\lambda)$. □

Lemma 7.24. Assuming index hiding property and somewhere argument of knowledge properties for seBARG_1, extraction correctness, consistency of extraction, for any QPT attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid 8** and **Hybrid 9**.

Proof. This follows from the proof of Lemma 7.14 with the edit that we use both $\mathsf{pT.vk}_1^\star$ and $\mathsf{pT.vk}_2^\star$ to check the extraction correctness property of seBARG_1. Therefore, there exists a negligible function $\mu_7(\lambda)$ s.t the advantage of attacker $\mathcal{A}$ in **Hybrid 9** is $\varepsilon(\lambda) - \mu_1(\lambda) - \mu_2(\lambda) - \mu_3(\lambda) - \mu_4(\lambda) - \mu_5(\lambda) - \mu_6(\lambda) - \mu_7(\lambda)$. □

Lemma 7.25. Assuming the soundness of the Commit-and-Measure Protocol (Maha) and the FHM protocol, **Hybrid 9** ($\mathfrak{R}'$) is a Commit-Challenge-Response protocol with computationally orthogonal projectors.

Proof. This follows from Lemma 4.10 where we replace the special index $i^\star$ with the index $j = j_1 = j_2 = j_3$ in the final round of the protocol. □

Lemma 7.26. The λ-fold repetition of **Hybrid 9**, i.e $(\mathfrak{R}')^\lambda$, has negligible soundness.

Proof. This follows from Theorem 3.27. □

Lemma 7.27. The λ-fold repetition of $\mathfrak{R}$, i.e $\mathfrak{R}^\lambda$, has negligible soundness.

Proof. The proof follows the proof of Lemma 4.12 by replacing seBARG index $i^\star$ with the index $j = j_1 = j_2 = j_3$ which is extracted at the final verification round. Therefore, we note that $\mathfrak{R}^\lambda$ satisfies the non-adaptive soundness property as defined in Definition 7.3 □

□

8 Public Coin Monotone Policy Batch Arguments for QMA

In this section, we outline a public coin monotone policy batch argument system for QMA. The protocol is public coin except for the verifier's first message, i.e. the setup phase. This property allows for round collapsing under the Quantum Random Oracle Model (see Section 9). More formally, this result can be stated as follows:

Theorem 8.1 (Designated Verifier Monotone Policy Batch Arguments for QMA)**.** Assuming the post-quantum hardness of the learning with errors problem (LWE), there is a **6-round** public-coin designated verifier interactive argument system for any monotone policy batch of k T-time quantum computations on inputs $x_1, \ldots, x_k$ over a function f (corresponding monotone circuit C_f) and corresponding $N = \mathsf{poly}(T)$ qubit witnesses $|\phi_1\rangle, \ldots, |\phi_k\rangle$, where

1. The Prover P is a QPT algorithm that runs in time $\mathsf{poly}(k, N)$
2. The Verifier V is a PPT algorithm that runs in time $\mathsf{poly}(\log k, N)$

3. The length of each message in the protocol is bounded by $\mathsf{poly}(\log k, N)$

The public coin protocol satisfies the completeness as defined in Definition 7.2 and a **Non-Adaptive** notion of soundness as defined in Definition 7.3.

8.1 Construction

Ingredients

1. Predicate Extractable Hash Family with Tags for bit-fixing predicates $\mathsf{PEHT}(\mathsf{Gen}, \mathsf{Hash}, \mathsf{Open}, \mathsf{Verify}, \mathsf{Extract}, \mathsf{ExtractIndex}, \mathsf{ExtractTag})$.
2. Fully Homomorphic Encryption or $\mathsf{FHE}(\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Enc}, \mathsf{Dec})$
3. ϵ-state preserving Arguments of Knowledge AoK as defined in Definition 3.16.
4. Commit-and-Measure Protocol $\mathsf{Maha}(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open}, \mathsf{Test}, \mathsf{Out})$ as defined in Definition 3.21.
5. The κ-fold FHM post hoc verification protocol $(\mathsf{P}_{\mathsf{FHM}}, \mathsf{V}_{\mathsf{FHM}})$ as defined in Definition 3.30 where κ is a polynomial in instance length.

Protocol Specification

- $\mathsf{Setup}(1^\lambda, 1^k, 1^N, f) \to (\mathsf{crs}, \mathsf{vk})$: Let C_f be the monotone circuit for the predicate f.
  - Setup the commit-and-measure protocol Maha. Sample randomness $r \xleftarrow{\$} \{0,1\}^\lambda$ and subsequently sample the basis string h using the randomness r.
    $$(\mathsf{mh.pk}, \mathsf{mh.sk}) = \mathsf{Maha.Gen}(1^\lambda; r)$$
  - Setup PEHT as follows for any random subset $J \subset [k]$ and $m = 0^J$
    $$(\mathsf{pT.hk}, \mathsf{pT.vk}, \mathsf{pT.td}) \leftarrow \mathsf{PEHT.Gen}(1^\lambda, k, J, m)$$
    We have the tag length for $(\mathsf{pT.hk}, \mathsf{pT.vk}, \mathsf{pT.td})$ being set to N.
  - Setup FHE as follows
    $$(\mathsf{fhe.pk}, \mathsf{fhe.sk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda)$$
    Compute the following
    $$\mathsf{ct}_r \leftarrow \mathsf{FHE.Enc}(\mathsf{fhe.pk}, r)$$
  - Sample the AoK hash key $\mathsf{aok.hk}$
  - Outputs the following.
    $$\mathsf{crs} = (\mathsf{mh.pk}, \mathsf{pT.hk}, \mathsf{pT.vk}, \mathsf{fhe.pk}, \mathsf{ct}_r, \mathsf{aok.hk})$$
    $$\mathsf{vk} = (\mathsf{fhe.sk})$$
- $\mathsf{Prove}(\mathsf{crs}, \{x_j\}_{j\in[k]}, \{|\phi_j\rangle^{\otimes\kappa}\}_{j\in[k]})$: Here, the quantum state $|\phi_i\rangle$ is the quantum witness to the statement x_i for some $\mathcal{L} \in \mathsf{QMA}$. The prover computes the string s s.t $s_i = (x_i \in \mathcal{L}_{\text{yes}})$ and proceeds as follows:

– **Round 1**: Parse crs to get $(\mathsf{pT.hk}, \mathsf{mh.pk})$. If $s_j = 0$, set $y_j = \perp$, else compute

$$|\psi_j\rangle \leftarrow \mathsf{P}_{\mathsf{FHM}}(x_j, |\phi_j\rangle^{\otimes\kappa})$$
$$(y_j, |\mathsf{st}_j\rangle) \leftarrow \mathsf{Maha.Com}(\mathsf{mh.pk}, |\psi_j\rangle)$$

Finally calculate h_y as follows and send to the verifier.

$$\mathsf{h}_y = \mathsf{PEHT.Hash}(\mathsf{pT.hk}, s, \{y_j\}_{j\in[k]})$$

– **Round 3**: Receive a bit $c \in \{0,1\}$ from the verifier. Parse crs to get $\mathsf{sB.crs}$. Let

$$z_j = \begin{cases} \perp & s_j = 0, \\ \mathsf{Maha.Open}(|\mathsf{st}_j\rangle, c) & s_j = 1 \end{cases}$$

Compute h_z as follows:

$$\mathsf{h}_z = \mathsf{PEHT.Hash}(\mathsf{pT.hk}, s, \{z_j\}_{j\in[k]})$$

Compute the evaluation of the circuit C_f as follows:

$$\forall j,\ \mathsf{ct}_j \leftarrow \mathsf{FHE.Eval}(\mathsf{fhe.pk}, \mathsf{VerifyInstance}(x_j, y_j, z_j, c, \cdot), \mathsf{ct}_r)$$
$$\mathsf{ct} \leftarrow \mathsf{FHE.Eval}(\mathsf{fhe.pk}, C_f(\cdot), \{\mathsf{ct}_j\}_{j\in[k]})$$

Send $(\mathsf{ct}, \mathsf{h}_z)$ to the verifier. Perform an AoK for the language $\mathcal{K}$ defined as follows:

$$\begin{aligned} \exists\ w \text{ s.t. } & \mathsf{h}_y = \mathsf{PEHT}(\mathsf{pT.hk}, s, \{y_j\}_{j\in[k]}) \wedge \\ & \mathsf{h}_z = \mathsf{PEHT}(\mathsf{pT.hk}, s, \{z_j\}_{j\in[k]}) \wedge \\ & \mathsf{ct}_j \leftarrow \mathsf{FHE.Eval}(\mathsf{fhe.pk}, \mathsf{VerifyInstance}(x_j, y_j, z_j, c, \cdot), \mathsf{ct}_r) \wedge \\ & \mathsf{ct} \leftarrow \mathsf{FHE.Eval}(\mathsf{fhe.pk}, C_f(\cdot), \{\mathsf{ct}_j\}_{j\in[k]}) \end{aligned}$$

where

$$w = \left(s, \{y_j\}_{j\in[k]}, \{z_j\}_{j\in[k]}, \{\mathsf{ct}_j\}_{j\in[k]}\right)$$

- $\mathsf{Verify}(\mathsf{crs}, \mathsf{vk}, \{x_j\}_{j\in[k]}, \mathsf{h}_y, \mathsf{h}_z, \mathsf{ct})$: Works as follows under normal (honest) execution.
  – **Round 2**: Receive h_y from the prover. Sample bit c and send it to the prover.
  – **Final Verification**: Receive $(\mathsf{ct}, \mathsf{h}_z)$ from the prover. Parse vk to get $\mathsf{fhe.sk}$. Check if

$$\mathsf{FHE.Dec}(\mathsf{fhe.sk}, \mathsf{ct}) = 1$$

If yes, perform an AoK for the language $\mathcal{K}$. Accept if AoK was successful.

Remark 8.2. We use $\mathfrak{G}$ to denote the above protocol and $\mathfrak{G}^\lambda$ to denote the λ-fold repetition of $\mathfrak{G}$.

8.2 Security Analysis

Completeness Completeness of $\mathfrak{R}$ follows immediately from the completeness properties of the underlying building blocks. For any monotone policy f for instances $x = (x_1, x_2, \ldots, x_k), C_f$ the interaction results in the verifier accepting with non-negligible probability if $x \in \mathcal{L}_f^{(k)}$. If $x_i \in \mathcal{L}_{\text{yes}}$, then $\mathsf{VerifyInstance}(x_i, \cdot, \cdot, \cdot, \cdot) = 1$ with non-negligible probability. The AoK interaction in the case of of an honest prover results in the prover being accepted. Therefore, if $x = (x_1, x_2, \ldots, x_k) \in \mathcal{L}_f^{(k)}$, the verifier accepts with high probability. The completeness of $\mathfrak{G}^\lambda$ is followed by a union-bound argument.

Round Complexity We observe that $\mathfrak{G}$ and (therefore) $\mathfrak{G}^\lambda$ both have **5 rounds** with setup. Alternatively, the verifier can do the setup and crs can be sent to the prover as the verifier's first message to get a **6 round Public Coin** designated verifier interactive monotone policy BARG for QMA.

Soundness The protocol $\mathfrak{G}^\lambda$ satisfies the notion of *non-adaptive computational soundness* as defined in definition 7.3. We now provide the soundness proof.

Theorem 8.3. Assuming the soundness of the Commit-and-Measure protocol (Definition 3.23) and the FHM protocol (Definition 3.32), computational binding and extraction correctness of tag and index of PEHT, and the extraction correctness of ϵ-state preserving AoK, the protocol $\mathfrak{G}^\lambda$ satisfies *non-adaptive computational soundness* as defined in Definition 7.3.

Proof. The proof strategy proceeds as follows. First, we note that $\mathfrak{R}$ is a *Commit-Challenge-Response* protocol as per Definition 3.24. For non-adaptive soundness, the challenger $\mathcal{C}$ and the attacker $\mathcal{A}$ are given the instance $x = (x_1, x_2, \ldots, x_k) \notin \mathcal{L}_f^{(k)}$ and the monotone circuit C_f pair (x, C_f). Additionally, the challenger $\mathcal{C}$ also receives the necessary set $J^\star \subset [k]$ for C_f as non-uniform advice. We first ensure that the hash values are consistent across **Round 1** and **Round 3**. Since AoK allows the full extraction of all witnesses, the challenger $\mathcal{C}$ can verify each instance and check if the root ciphertext was computed correctly. Finally, we show that the final hybrid has a negligible advantage over the original protocol. From this we follow the arguments presented in Section 4 to show that the protocol $\mathfrak{G}^\lambda$ has negligible soundness. We proceed as follows

Hybrid 0: Original Protocol

Hybrid 1: Same as the previous execution, except that we now enable extraction on the necessary set $J^\star$ for PEHT, i.e.,

$$(\mathsf{pT.ht}^\star, \mathsf{pT.vk}^\star, \mathsf{pT.td}^\star) \leftarrow \mathsf{PEHT.Gen}(1^\lambda, k, J^\star, 0^{J^\star})$$

Hybrid 2: Enable AoK full-extraction, i.e. extract the witness as $(\tilde{\tau}^\star, \omega^\star) \leftarrow E^{\mathcal{A}(\cdot,\cdot)}$

Hybrid 3: We have the following structure for the extracted witness

$$\omega^\star = \left(s^\star, s', \{y_j^\star, z_j^\star, \mathsf{ct}_j^\star\}_{j\in[k]}\right)$$

Reject if any of the following checks fail

$$\begin{aligned} \mathsf{h}_y &= \mathsf{PEHT.Hash}(\mathsf{pT.hk}^\star, s^\star, \{y_j^\star\}_{j\in[k]}) \wedge \\ \mathsf{h}_z &= \mathsf{PEHT.Hash}(\mathsf{pT.hk}^\star, s', \{z_j^\star\}_{j\in[k]}) \wedge \\ \forall j,\ \mathsf{ct}_j^\star &= \mathsf{FHE.Eval}(\mathsf{fhe.pk}, \mathsf{VerifyInstance}(x_j, y_j^\star, z_j^\star, c, \cdot), \mathsf{ct}_r) \ \wedge s^\star = s' \end{aligned}$$

Hybrid 4: We now update the protocol by extracting the tags and indices in final verification round. Reject if $j_1, j_2 \notin J^\star$

$$\begin{aligned} j_1 &= \mathsf{PEHT.ExtractIndex}(\mathsf{pT.td}^\star, \mathsf{h}_y) \quad y'_{j_1} = \mathsf{PEHT.ExtractTag}(\mathsf{pT.td}^\star, \mathsf{h}_y) \\ j_2 &= \mathsf{PEHT.ExtractIndex}(\mathsf{pT.td}^\star, \mathsf{h}_z) \quad z'_{j_2} = \mathsf{PEHT.ExtractTag}(\mathsf{pT.td}^\star, \mathsf{h}_z) \end{aligned}$$

Hybrid 5: Replace the $\mathsf{FHE.Dec}$ for ct by performing the following check

$$\forall j,\ b_j = \mathsf{VerifyInstance}(x_j, y_j^\star, z_j^\star, c, r)$$
$$C_f(b_1, b_2, \ldots, b_k) = 1$$

Hybrid 6: We now replace ct_r in the crs with $\mathsf{ct}_0 = \mathsf{FHE.Enc}(\mathsf{fhe.pk}, 0^\lambda)$ and let the protocol continue as in the previous hybrid.

We now prove that the attacker's advantage changes negligibly through the sequence of hybrids assuming the soundness properties of the cryptographic building blocks being used. Subsequently, we show that **Hybrid 6** is a Commit-Challenge-Response protocol that has orthogonal projectors. We then show that since the attacker's advantage has changed negligibly from **Hybrid 0** to **Hybrid 6**, $\mathfrak{G}^\lambda$ cannot have non-negligible soundness. To begin, let us assume that the attacker's advantage in breaking the non-adaptive soundness property of **Hybrid 0** is $\delta(\lambda)$.

Lemma 8.4. Assuming the predicate hiding property of PEHT, for any QPT attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid 0** and **Hybrid 1**.

Proof. This proof follows from the proof of Lemma 7.11. Therefore, there exists a negligible function $\mu_1(\lambda)$ s.t the advantage of the attacker $\mathcal{A}$ in **Hybrid 1** is $\delta(\lambda) - \mu_1(\lambda)$ □

Lemma 8.5. Assuming that the AoK is an ϵ-state preserving AoK, for any QPT attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid 1** and **Hybrid 2**.

Proof. An attacker $\mathcal{A}$ with a non-negligible advantage across **Hybrid 1** and **Hybrid 2** can differentiate between the output distributions of *Real* and *Simulated* games of the AoK, thus contradicting the ϵ-state preserving nature of AoK. Therefore, there exists a negligible function $\mu_2(\lambda)$ s.t the advantage of the attacker $\mathcal{A}$ in **Hybrid 2** is $\delta(\lambda) - \mu_1(\lambda) - \mu_2(\lambda)$. □

Lemma 8.6. Assuming the extraction correctness of the ϵ-state preserving AoK, for any QPT attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid 2** and **Hybrid 3**.

Proof. An attacker $\mathcal{A}$ with a non-negligible advantage across **Hybrid 2** and **Hybrid 3** must be able to produce an accepting transcript $\tilde{\tau}^\star$ with witness $\omega^\star$ s.t

$$\mathsf{h}_y \neq \mathsf{PEHT.Hash}(\mathsf{pT.hk}^\star, s^\star, \{y_j^\star\}_{j\in[k]}) \vee$$
$$\mathsf{h}_z \neq \mathsf{PEHT.Hash}(\mathsf{pT.hk}^\star, s', \{z_j^\star\}_{j\in[k]}) \vee$$
$$\forall j,\ \mathsf{ct}_j^\star \neq \mathsf{FHE.Eval}(\mathsf{fhe.pk}, \mathsf{VerifyInstance}(x_j, y_j^\star, z_j^\star, c, \cdot), \mathsf{ct}_r) \ \vee s^\star \neq s'$$

However, this contradicts the Extraction correctness property of the ϵ-state preserving AoK. Therefore, there exists a negligible function $\mu_3(\lambda)$ s.t the advantage of the attacker $\mathcal{A}$ in **Hybrid 3** is $\delta(\lambda) - \mu_1(\lambda) - \mu_2(\lambda) - \mu_3(\lambda)$. □

Lemma 8.7. Assuming the index extraction correctness of PEHT, for any QPT attacker $\mathcal{A}$, the attacker's advantage doesn't change in **Hybrid 3** and **Hybrid 4**

Proof. This follows the proof of Lemma 7.13. Therefore, the advantage of the attacker $\mathcal{A}$ in **Hybrid 4** is $\delta(\lambda) - \mu_1(\lambda) - \mu_2(\lambda) - \mu_3(\lambda)$. □

Lemma 8.8. Assuming correctness of FHE evaluation, for any QPT-attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid 4** and **Hybrid 5**.

Proof. An attacker $\mathcal{A}$ with non-negligible advantage across **Hybrid 4** and **Hybrid 5** must imply

$$\mathsf{FHE.Dec}(\mathsf{fhe.sk}, \mathsf{ct}) = 1 \ \wedge \ C_f(b_1, b_2, \ldots, b_k) = 0$$

a contradiction to the correctness of FHE evaluation. Therefore, there exists a negligible function $\mu_4(\lambda)$ s.t the advantage of the attacker $\mathcal{A}$ in **Hybrid 5** is $\delta(\lambda) - \mu_1(\lambda) - \mu_2(\lambda) - \mu_3(\lambda) - \mu_4(\lambda)$. □

Lemma 8.9. Assuming the semantic security of the FHE scheme, for every QPT attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid 5** and **Hybrid 6**.

Proof. This proof follows from Lemma 5.16 where we update the security game to $\mathcal{B}$ send $(r, 0^\lambda)$ and run the verification as per **Hybrid 5**. Thus, there exists a negligible function $\mu_5(\lambda)$ s.t the advantage of the attacker $\mathcal{A}$ in **Hybrid 6** is $\delta(\lambda) - \mu_1(\lambda) - \mu_2(\lambda) - \mu_3(\lambda) - \mu_4(\lambda) - \mu_5(\lambda)$. □

Lemma 8.10. Assuming the soundness of the Commit-and-Measure Protocol (Maha) and the FHM protocol, **Hybrid 6** ($\mathfrak{G}'$) is a Commit-Challenge-Response protocol with computationally orthogonal projectors.

Proof. This follows from Lemma 4.10 where we replace the special index $i^\star$ with the index $j_1 \in J^\star$ in the final round of the protocol. □

Lemma 8.11. The λ-fold repetition of **Hybrid 6**, i.e $(\mathfrak{G}')^\lambda$, has negligible soundness.

Proof. This follows from Theorem 3.27. □

Lemma 8.12. The λ-fold repetition of $\mathfrak{G}$, i.e $\mathfrak{G}^\lambda$, has negligible soundness.

Proof. The proof follows the proof of Lemma 4.12 by replacing the seBARG index $i^\star$ with the index $j_1 \in J^\star$. Therefore, we note that $\mathfrak{G}^\lambda$ satisfies the non-adaptive soundness property as defined in Definition 7.3. □

□

9 Non-Interactive Batch Arguments in QROM

This section discusses the construction of non-interactive batch arguments in QROM starting from the public coin BARG presented in Section 5. In [DFMS19], the authors showed how to reprogram a QROM adaptively on a single input. Consequently, this led to the conclusion that the Fiat-Shamir transformation of a Σ-protocol is as secure in the QROM as in the standard model. Following this, in [DFM20], the results were extended to show the re-programmability of QROM on multiple inputs to establish the security of Fiat-Shamir transformation of *multi-round* public coin interactive protocols in the QROM. We state this result as the following theorem.

Theorem 9.1 (Measure and Reprogram with multiple inputs and enforced extraction order [DFM20])**.** Let n be a positive integer, and let $\mathcal{X}_0, \mathcal{X}$, and $\mathcal{Y}$ be finite non-empty sets. A black-box polynomial time $(n+1)$-stage quantum algorithm $\mathcal{S}$ exists, satisfying the following property. Let $\mathcal{A}$ be an arbitrary quantum oracle algorithm that makes q queries to a uniformly random $H : (\mathcal{X}_0 \cup \mathcal{Y}) \times \mathcal{X} \to \mathcal{Y}$ and that outputs a tuple $\mathbf{x} = (x_0, \ldots, x_n) \in (\mathcal{X} \cup \mathcal{X}_0)$ and a (possibly quantum) output z. Then, for any $\mathbf{x}^o \in (\mathcal{X} \cup \mathcal{X}_0)$ without duplicate entries and for any predicate V:

$$\begin{aligned}&\Pr_\Theta\left[\mathbf{x} = \mathbf{x}^o \wedge V(x, \Theta, z) \ : \ (\mathbf{x}, z) \leftarrow \langle \mathcal{S}^{\mathcal{A}}, \Theta\rangle\right] \\ &\geq \frac{n!}{(2q+n+1)^{2n}} \Pr_H\left[\mathbf{x} = \mathbf{x}^o \wedge V(\mathbf{x}, \mathbf{h}^{H,\mathbf{x}}, z) : (x, z) \leftarrow \mathcal{A}^H\right] - \epsilon_{\mathbf{x}^o}\end{aligned}$$

where $\mathbf{h}^{H,\mathbf{x}} = (h_1^{H,\mathbf{x}}, \ldots, h_n^{H,\mathbf{x}})$ denotes the hash chain defined as $h_1^{H,\mathbf{x}} = H(x_0, x_1)$ and $h_i^{H,\mathbf{x}} = H(h_{i-1}^{H,\mathbf{x}}, x_i)$ for $1 \leq i \leq n$, and $\epsilon_{\mathbf{x}^o}$ is equal to $\dfrac{n!}{|\mathcal{Y}|}$ when summed over all $\mathbf{x}^o$.

We are now ready to show the main result which is obtained by applying the Fiat-Shamir transformation to the designated verifier public coin interactive batch argument system constructed in Section 5 and Section 8.

Theorem 9.2. If a constant round public coin (except for the first message) batch protocol (P, V) satisfies non-adaptive soundness, then the result of application of Fiat-Shamir transform on it is the protocol (P_{FS}, V_{FS}) which also satisfies non-adaptive soundness.

Proof. Suppose the protocol (P, V) has $2n+1$ rounds and since it is public coin except for the first message, the verifier can be considered to consist of two circuits (V_0, V_1). Let the circuit C_f be the monotone circuit defining the language $\mathcal{L}_f^{(k)}$. Given a batch of instances $\mathbf{x} = (x_1, \ldots, x_k, C_f)$, the verifier uses random coins r to compute the first message as $s_0 = V_0(\mathbf{x}, r)$. The remaining verifier messages are random strings $s_1, \ldots, s_n$. We denote the prover messages as $t_0, \ldots, t_n$. Finally, the verifier outputs the result of circuit $V_1(\mathbf{x}, r, s_0, t_0, s_1, t_1, \ldots, s_n, t_n)$.

Let us now consider a random oracle $H : ((\mathcal{X} \times \mathsf{S}) \cup \mathsf{S}) \times ([n] \times \mathcal{T}) \to \mathsf{S}$, where $\mathcal{X}$ is the space of (batch of) instances, S is the space of verifier messages, and $\mathcal{T}$ is the space of prover messages. (P_{FS}, V_{FS}) proceeds as follows. First, V_{FS} outputs $s_0 = V_0(\mathbf{x}, r)$. Next, the prover produces $t_0 = P(\mathbf{x}, s_0)$. Subsequently, it generates remaining verifier messages as $s_i = H((\mathbf{x}, s_0), (i, t_{i-1}))$ and prover messages as $t_i = P(\mathbf{x}, s_i)$. Finally, the verifier verifies the correctness of each s_i and outputs $V(\mathbf{x}, r, s_0, t_0, \ldots, s_n, t_n)$.

Consider an instance $\mathbf{x}$ such that $C_f(x_1, x_2, \ldots x_k) \neq 1$, i.e $x_1, x_2, \ldots, x_k \notin \mathcal{L}_f^{(k)}$. Let $\mathcal{A}$ be an adversary that is initialised with $(\mathbf{x}, s_0)$ and $\epsilon(r)$ be its success probability defined as follows. Note that it can generate $s_1, \ldots, s_n$ using no additional information.

$$\epsilon(r) = \Pr_H\left[(\mathbf{x}, s_0, t_0, s_1, t_1, \ldots, s_n, t_n) = 1 \ : \ (t_0, \ldots, t_n) \leftarrow \mathcal{A}^H\right]$$

We make use of Theorem 9.1 to get the following bound, where $\delta(r)$ is the success probability of a simulator $\mathcal{S}^{\mathcal{A},r}(s_1, \ldots, s_n)$.

$$\delta(r) \geq \frac{1}{\mathsf{poly}(\lambda)}\epsilon(r) - \mathsf{negl}(\lambda)$$

But, due to the non-adaptive soundness of (P, V) it must be the case that $\mathbb{E}_r[\delta(r)] = \mathsf{negl}(\lambda)$. Therefore, we can write the following.

$$\mathsf{negl}(\lambda) \geq \frac{1}{\mathsf{poly}(\lambda)}\mathbb{E}_r[\epsilon(r)] - \mathsf{negl}(\lambda)$$
$$\implies \epsilon(r) = \mathsf{negl}(\lambda)$$

□

Corollary 9.3 (Designated verifier Non-Interactive Monotone Policy BARG for QMA in QROM)**.** Assuming the post-quantum hardness of learning with errors (LWE) problem and post-quantum fully homomorphic encryption (FHE), there exists a designated verifier non-interactive monotone policy batch argument system for QMA in the quantum random oracle model (QROM) which satisfies the completeness (Definition 7.2) and non-adaptive soundness properties (Definition 7.3)

Theorem 9.4. If a constant round public coin (except for the first message) batch protocol (P, V) satisfies semi-adaptive soundness, then the result of application of Fiat-Shamir transform on it is the protocol (P_{FS}, V_{FS}) which also satisfies semi-adaptively sound.

Proof. The proof of this theorem proceeds similarly to Theorem 9.2 with slight modifications to incorporate the semi-adaptive soundness nature of the batch protocol (P, V). The first message s_0 is generated with the additional input of $i^\star$. Finally, we invoke the semi-adaptive soundness property of the batch argument (P, V) to conclude that (P_{FS}, V_{FS}) is also semi-adaptively sound. □

Corollary 9.5 (Designated Verifier Non-Interactive BARG for QMA in QROM)**.** Assuming the post-quantum hardness of learning with errors (LWE) problem and post-quantum fully homomorphic encryption (FHE), there exists a non-interactive batch argument system for QMA in the quantum random oracle model (QROM) which satisfies the completeness and semi-adaptive computational soundness properties defined in Section 3.20.

10 Public Verifiability using Differing inputs iO

In this section, we outline a canonical construction where we transform any designated verifier proof system to a publicly verifiable proof system. For simplicity, we show the construction for a generic privately verifiable non-interactive $\mathsf{SNARG}(\mathsf{Gen}, \mathsf{P}, \mathsf{Ver})$ system for a given language $\mathcal{L}$. More formally, this result can be stated as follows:

Theorem 10.1 (Publicly verifiable proof system)**.** Assuming the security of Differing Inputs Obfuscation $\mathsf{di}\mathcal{O}$, Puncturable PRFs, and the underlying privately verifiable proof system $\mathsf{SNARG}(\mathsf{Gen}, \mathsf{P}, \mathsf{V})$, there exists a publicly verifiable proof system $\mathsf{pvSNARG}(\widetilde{\mathsf{Gen}}, \widetilde{\mathsf{P}}, \widetilde{\mathsf{V}})$ that satisfies a **non-adaptive** notion of soundness.

10.1 Construction

We construct the $\mathsf{pvSNARG}(\widetilde{\mathsf{Gen}}, \widetilde{\mathsf{P}}, \widetilde{\mathsf{V}})$ by obfuscating the two programs: one to generate the prover's key pk, and one to verify the instance x given proof π. We rely on PPRF security and $\mathsf{di}\mathcal{O}$ security to achieve a non-adaptive notion of soundness. However, we believe this transformation can be extended to get somewhere soundness by changing the verification program to output 0 for the semi-adaptively chosen instance $i^\star$.

Ingredients

1. Privately verifiable $\mathsf{SNARG}(\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ proof system.
2. Differing inputs Obfuscation $\mathsf{di}\mathcal{O}$ as defined in Definition 3.14.
3. Puncturable PRF family $\mathsf{PPRF}(\mathsf{Eval}, \mathsf{Puncture}, \mathsf{PuncEval})$ with output length $\ell(\lambda)$ as defined in Definition 3.9.

Protocol Specification

- $\widetilde{\mathsf{Gen}}(1^\lambda, x) \to \mathsf{crs}$: It first samples a PPRF key $K \leftarrow \{0,1\}^\lambda$, and is hard-wired in the following procedures.

$P_{\mathsf{pk}}(x)$:

1. Compute $(\mathsf{pk}_x, \mathsf{vk}_x) = \mathsf{SNARG.Gen}(1^\lambda, F_K(x))$
2. Output pk_x.

$P_{\mathsf{V}}(x, \pi)$:

1. Compute $(\mathsf{pk}_x, \mathsf{vk}_x) = \mathsf{SNARG.Gen}(1^\lambda, F_K(x))$
2. Run $b_{x,\pi} = \mathsf{SNARG.V}(\mathsf{vk}_x, x, \pi)$
3. Output $b_{x,\pi}$.

  Compute the obfuscation of the above procedures

$$\widetilde{P}_{\mathsf{pk}} \leftarrow \mathsf{di}\mathcal{O}(1^\lambda, P_{\mathsf{pk}}), \quad \widetilde{P}_{\mathsf{V}} \leftarrow \mathsf{di}\mathcal{O}(1^\lambda, P_{\mathsf{V}})$$

  Output $\mathsf{crs} = (\widetilde{P}_{\mathsf{pk}}, \widetilde{P}_{\mathsf{V}})$.

- $\widetilde{\mathsf{P}}(\mathsf{crs}, x, w) \to \pi$: Parse crs to get $\widetilde{P}_{\mathsf{pk}}$. Perform the following steps and output π:

$$\begin{aligned} \mathsf{pk}_x &= \widetilde{P}_{\mathsf{pk}}(x) \\ \pi &\leftarrow \mathsf{SNARG.P}(\mathsf{pk}_x, x, w) \end{aligned}$$

- $\widetilde{\mathsf{V}}(\mathsf{crs}, x, \pi) \to 0/1$: Parse crs to get $\widetilde{P}_{\mathsf{V}}$. Output $\widetilde{P}_{\mathsf{V}}(x, \pi)$

10.2 Security Analysis

Completeness The completeness of the protocol follows directly from the underlying cryptographic primitives. The obfuscated program $\widetilde{P}_{\mathsf{V}}$ accepts with overwhelming probability whenever $\mathsf{SNARG.V}$ accepts. Therefore if $x \in \mathcal{L}$, the verifier accepts with high probability.

Soundness The protocol $\mathsf{pvSNARG}(\widetilde{\mathsf{Gen}}, \widetilde{\mathsf{P}}, \widetilde{\mathsf{V}})$ satisfies the notion of non-adaptive soundness. We now provide the soundness proof.

Theorem 10.2. Assuming the security of Differing Inputs Obfuscation $\mathsf{di}\mathcal{O}$, Pucturable PRFs, and the non-adaptive soundness of the underlying SNARG proof system, the protocol $\mathsf{pvSNARG}(\widetilde{\mathsf{Gen}}, \widetilde{\mathsf{P}}, \widetilde{\mathsf{V}})$ satisifies non-adaptive computational soundness.

Proof. The proof strategy is as follows. For non-adaptive soundness, we are given an instance $x^\star \notin \mathcal{L}$. First we make the PPRF puncturable on this instance $x^\star$. We then proceed to update the $\widetilde{P}_{\mathsf{pk}}$ to an alternative program that outputs a special pk for $x^\star$. From here, we update the verification program to output 0 on input $x^\star$. If at any point, the attacker $\mathcal{A}$ has non-negligible advantage, we contradict the security of $\mathsf{di}\mathcal{O}$ or Puncturable PRFs. We proceed as follows

Hybrid 0: Same as the original protocol

Hybrid 1: Sample $\mathsf{pk}_{x^\star}$ as follows:

$$(\mathsf{pk}_{x^\star}, \mathsf{vk}_{x^\star}) = \mathsf{SNARG.Gen}(1^\lambda, F_K(x^\star))$$

Hardwire $\mathsf{pk}_{x^\star}$, puncture PPRF at $x = x^\star$ and obfuscate the following programs in the crs

$P'_{\mathsf{pk}}(x)$:

Hardwired values: $\mathsf{pk}_{x^\star}$

(a) If $x = x^\star$, output $\mathsf{pk}_{x^\star}$.

(b) Else:

i. Compute $(\mathsf{pk}_x, \mathsf{vk}_x) = \mathsf{SNARG.Gen}(1^\lambda, F_{K\{x^\star\}}(x))$.

ii. Output pk_x

$P'_{\mathsf{V}}(x, \pi)$:

Hardwired values: $\mathsf{vk}_{x^\star}$

(a) If $x = x^\star$ output $\mathsf{SNARG.V}(\mathsf{vk}_{x^\star}, x, \pi)$.

(b) Else:

i. Compute $(\mathsf{pk}_x, \mathsf{vk}_x) = \mathsf{SNARG.Gen}(1^\lambda, F_{K\{x^\star\}}(x))$

ii. Output $\mathsf{SNARG.V}(\mathsf{vk}_x, x, \pi)$.

Hybrid 2: Generate the keys for the instance $x^\star$ using fresh randomness $r \leftarrow \{0,1\}^{\ell(\lambda)}$,

$$(\mathsf{pk}_{x^\star}, \mathsf{vk}_{x^\star}) = \mathsf{SNARG.Gen}(1^\lambda, r)$$

We obfuscate the following in the crs

$P''_{\mathsf{pk}}(x)$:

Hardwired values: $\mathsf{pk}_{x^\star}$

(a) If $x = x^\star$, output $\mathsf{pk}_{x^\star}$.

(b) Else:

i. Compute $(\mathsf{pk}_x, \mathsf{vk}_x) = \mathsf{SNARG.Gen}(1^\lambda, F_{K\{x^\star\}}(x))$.

ii. Output pk_x

$P''_{\mathsf{V}}(x, \pi)$:

Hardwired values: $\mathsf{vk}_{x^\star}$

(a) If $x = x^\star$ output $\mathsf{SNARG.V}(\mathsf{vk}_{x^\star}, x, \pi)$.

(b) Else:

i. Compute $(\mathsf{pk}_x, \mathsf{vk}_x) = \mathsf{SNARG.Gen}(1^\lambda, F_{K\{x^\star\}}(x))$

ii. Output $\mathsf{SNARG.V}(\mathsf{vk}_x, x, \pi)$.

Hybrid 3: We obfuscate the following program in the crs

$P'''_{\mathsf{V}}(x, \pi)$:

(a) If $x = x^\star$, output 0.

(b) Else

i. Compute $(\mathsf{pk}_x, \mathsf{vk}_x) = \mathsf{SNARG.Gen}(1^\lambda, F_{K\{x^\star\}}(x))$

ii. Output $\mathsf{SNARG.V}(\mathsf{vk}_x, x, \pi)$

For contradiction, assume that the advantage of the attacker $\mathcal{A}$ in breaking the non-adaptive soundness property is $\varepsilon(\lambda)$, a non-negligible function. We have the following lemmas

Lemma 10.3. Assuming the security of differing inputs obfuscation $\mathsf{di}\mathcal{O}$, for any poly-size attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid 0** and **Hybrid 1**.

Proof. First, we observe that the input/output behavior of $P_{\mathsf{pk}}(x)$ and P'_{pk} is identical. Similarly, we observe that P_{V} and P'_{V} are identical. We only puncture the randomness to the $\mathsf{SNARG.Gen}$ algorithm at $x^\star$. For contradiction, assume that $\mathcal{A}$ has non-negligible advantage across **Hybrid 0** and **Hybrid 1**. Consider a reduction algorithm $\mathcal{B}$ that receives an obfuscated program $\widetilde{P}_{\mathsf{pk}}$ from a challenger $\mathcal{C}$ that samples at random one of P_{pk} or P'_{pk} with aux set to any additional public information that is generated by the $\mathsf{SNARG.Gen}$. We note that if P_{pk} is chosen, we are operating in **Hybrid 0**, while if P'_{pk} is chosen, we are operating in **Hybrid 1**. If $\mathcal{A}$ has non-negligible advantage between the two hybrids, the reduction algorithm $\mathcal{B}$ can guess which program was used under obfuscation. This contradicts the security of $\mathsf{di}\mathcal{O}$. Therefore, there exists a negligible function $\mu_1(\lambda)$ s.t the advantage of the attacker $\mathcal{A}$ in **Hybrid 1** is $\varepsilon(\lambda) - \mu_1(\lambda)$. □

Lemma 10.4. Assuming the security of the PPRF, for any poly-size attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid 1** and **Hybrid 2**.

Proof. First, we observe that the only change between **Hybrid 1** and **Hybrid 2** is the randomness used to generate the $(\mathsf{pk}, \mathsf{vk})$ pairs for the adaptively chosen instance $x^\star$ in the programs P'_{pk} and P''_{pk} (similarly for P'_{V} and P''_{V}). For contradiction, assume that $\mathcal{A}$ has non-negligible advantage across the two hybrids. We can construct a reduction algorithm $\mathcal{B}$ that uses $\mathcal{A}$. The algorithm $\mathcal{B}$ interacts with $\mathcal{A}$ trivially to produce the output $(F_{K\{x^\star\}}(x), r)$ and successfully differentiates between the two with non-negligible probability, a contradiction to the security of PPRFs. Therefore, there exists a negligible function $\mu_2(\lambda)$ s.t the advantage of the attacker $\mathcal{A}$ in **Hybrid 2** is $\varepsilon(\lambda) - \mu_1(\lambda) - \mu_2(\lambda)$. □

Lemma 10.5. Assuming the security of Differing Inputs Obfuscation $\mathsf{di}\mathcal{O}$, for any poly-size attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid 2** and **Hybrid 3**.

Proof. First, we observe that the input/output behavior of P''_{V} and P'''_{V} differs only on the input $x^\star$. For all $x \neq x^\star$, both P''_{V} and P'''_{V} must behave exactly the same. Note that P'''_{V} is well defined since we are quantifying over all $x^\star$, $x^\star$ must be known to the reduction algorithm. For contradiction, assume that $\mathcal{A}$ has non-negligible advantage across **Hybrid 2** and **Hybrid 3**. Since non-negligible advantage is greater than inverse polynomial advantage, we can invoke extraction on the attacker $\mathcal{A}$. Let the extraction output $\mathcal{E}^{\mathcal{A}}$ be x'. If $x^\star \neq x'$, we break the security of $\mathsf{di}\mathcal{O}$. Therefore, there exists a negligible function $\mu_3(\lambda)$ s.t the advantage of the attacker $\mathcal{A}$ in **Hybrid 3** is $\varepsilon(\lambda) - \mu_1(\lambda) - \mu_2(\lambda) - \mu_3(\lambda)$. □

At this point, the advantage of the attacker is $\varepsilon(\lambda) - \mathsf{negl}(\lambda)$, where

$$\mathsf{negl}(\lambda) = (\mu_1 + \mu_2 + \mu_3)(\lambda)$$

If $x^\star = x'$, the challenger $\mathcal{C}$ performs the following steps

$$(\mathsf{pk}_{x^\star}, \mathsf{vk}_{x^\star}) \leftarrow \mathsf{SNARG.Gen}(1^\lambda, r)$$
$$b \leftarrow \mathsf{SNARG.V}(\mathsf{vk}_{x^\star}, x^\star, \pi)$$

If $b = 1$, we break the non-adaptive soundness of the underlying SNARG proof system, a contradiction. Therefore, the attacker's advantage in **Hybrid 0** must be negligible, i.e $\varepsilon(\lambda) = \mathsf{negl}(\lambda)$ □

Acknowledgements. We thank Hayden Coffey for helpful discussions and collaboration during early stages of this work. We also thank Dakshita Khurana for insightful discussions.

References

[ABG+13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. *Cryptology ePrint Archive*, 2013.

[ACGH20] Gorjan Alagic, Andrew M Childs, Alex B Grilo, and Shih-Han Hung. Non-interactive classical verification of quantum computation. In *Theory of Cryptography Conference*, pages 153–180. Springer, 2020.

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 308–326, 2015.

[AJS15] Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Achieving compactness generically: Indistinguishability obfuscation from non-compact functional encryption. *IACR Cryptology ePrint Archive*, 2015.

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, May 1998.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of np. *Journal of the ACM (JACM)*, 45(1):70–122, 1998.

[BBK+23] Zvika Brakerski, Maya Farber Brodsky, Yael Tauman Kalai, Alex Lombardi, and Omer Paneth. Snargs for monotone policy batch np. In *Annual International Cryptology Conference*, pages 252–283. Springer, 2023.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of computer and system sciences*, 37(2):156–189, 1988.

[BCC+17] Nir Bitansky, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Huijia Lin, Aviad Rubinstein, and Eran Tromer. The hunting of the SNARK. *Journal of Cryptology*, 30(4):989–1066, 2017.

[BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 326–349, 2012.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 111–120, 2013.

[BCI+13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In *Theory of Cryptography: 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*, pages 315–333, 2013.

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In *Theory of cryptography conference*, pages 52–73. Springer, 2014.

[BDGM20] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Factoring and pairings are not necessary for io: Circular-secure lwe suffices. *Cryptology ePrint Archive*, 2020.

[BFL91] László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational complexity*, 1:3–40, 1991.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, pages 1–18, 2001.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In *International workshop on public key cryptography*, pages 501–519. Springer, 2014.

[BGMZ18] James Bartusek, Jiaxin Guan, Fermi Ma, and Mark Zhandry. Return of ggh15: provable security against zeroizing attacks. In *Theory of Cryptography Conference*, pages 544–574. Springer, 2018.

[BHK17a] Zvika Brakerski, Justin Holmgren, and Yael Kalai. Non-interactive delegation and batch np verification from standard computational assumptions. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 474–482, 2017.

[BHK17b] Zvika Brakerski, Justin Holmgren, and Yael Tauman Kalai. Non-interactive delegation and batch NP verification from standard computational assumptions. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 474–482, 2017.

[BKK+18] Saikrishna Badrinarayanan, Yael Tauman Kalai, Dakshita Khurana, Amit Sahai, and Daniel Wichs. Succinct delegation for low-space non-deterministic computation. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 709–721, 2018.

[BKL+22] James Bartusek, Yael Tauman Kalai, Alex Lombardi, Fermi Ma, Giulio Malavolta, Vinod Vaikuntanathan, Thomas Vidick, and Lisa Yang. Succinct classical verification of quantum computation. In *Annual International Cryptology Conference*, pages 195–211. Springer, 2022.

[BL08] Jacob D Biamonte and Peter J Love. Realizable hamiltonians for universal adiabatic quantum computers. *Physical Review A*, 78(1):012352, 2008.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: definitions and challenges. In *Proceedings of the 8th conference on Theory of cryptography*, TCC'11, pages 253–273, Berlin, Heidelberg, 2011. Springer-Verlag.

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. *SIAM Journal on computing*, 43(2):831–871, 2014.

[BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 171–190, 2015.

[BV17] Nir Bitansky and Vinod Vaikuntanathan. A note on perfect correctness by derandomization. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 592–606. Springer, 2017.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *Advances in Cryptology-ASIACRYPT 2013: 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part II 19*, pages 280–300. Springer, 2013.

[BWW24] Eli Bradley, Brent Waters, and David J. Wu. Batch arguments to NIZKs from one-way functions. In *TCC*, 2024.

[CCY20] Nai-Hui Chia, Kai-Min Chung, and Takashi Yamakawa. Classical verification of quantum computations with efficient verifier. In *Theory of Cryptography Conference*, pages 181–206. Springer, 2020.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *Journal of the ACM (JACM)*, 51(4):557–594, 2004.

[CGKS23] Matteo Campanelli, Chaya Ganesh, Hamidreza Khoshakhlagh, and Janno Siim. Impossibilities in succinct arguments: Black-box extraction and more. In *International Conference on Cryptology in Africa*, pages 465–489. Springer, 2023.

[CJJ21] Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. Non-interactive batch arguments for np from standard assumptions. In *Annual International Cryptology Conference*, pages 394–423. Springer, 2021.

[CJJ22] Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. Snargs for p from lwe. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 68–79. IEEE, 2022.

[CLLZ21] Andrea Coladangelo, Jiahui Liu, Qipeng Liu, and Mark Zhandry. Hidden cosets and applications to unclonable cryptography. In *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part I 41*, pages 556–584. Springer, 2021.

[CM16] Toby Cubitt and Ashley Montanaro. Complexity classification of local hamiltonian problems. *SIAM Journal on Computing*, 45(2):268–316, 2016.

[CMSZ22] Alessandro Chiesa, Fermi Ma, Nicholas Spooner, and Mark Zhandry. Post-quantum succinct arguments: breaking the quantum rewinding barrier. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 49–58. IEEE, 2022.

[CVW18] Yilei Chen, Vinod Vaikuntanathan, and Hoeteck Wee. Ggh15 beyond permutation branching programs: proofs, attacks, and candidates. In *Advances in Cryptology–CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part II 38*, pages 577–607. Springer, 2018.

[DFM20] Jelle Don, Serge Fehr, and Christian Majenz. The measure-andreprogram technique 2.0: multi-round fiat-shamir and more. In *Annual International Cryptology Conference*, pages 602–631, 2020.

[DFMS19] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Security of the fiat-shamir transformation in the quantum random-oracle model. In *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part II 39*, pages 356–383. Springer, 2019.

[DQV+21] Lalita Devadas, Willy Quach, Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Succinct lwe sampling, random polynomials, and obfuscation. In *Theory of Cryptography: 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8–11, 2021, Proceedings, Part II 19*, pages 256–287. Springer, 2021.

[FHM18] Joseph F. Fitzsimons, Michal Hajduš ek, and Tomoyuki Morimae. Post hoc verification of quantum computation. *Physical Review Letters*, 120(4), jan 2018.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the theory and application of cryptographic techniques*, pages 186–194. Springer, 1986.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178, 2009.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM (JACM)*, 33(4):792–807, 1986.

[GKNV24] Sam Gunn, Yael Tauman Kalai, Anand Natarajan, and Agi Villanyi. Classical commitments to quantum states, 2024.

[Goy24] Rishab Goyal. Mutable batch arguments and applications. *Cryptology ePrint Archive*, 2024.

[GP21] Romain Gay and Rafael Pass. Indistinguishability obfuscation from circular security. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 736–749, 2021.

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In *Advances in Cryptology-ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings 16*, pages 321–340, 2010.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35*, pages 305–326, 2016.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 99–108, 2011.

[GZ20] Marios Georgiou and Mark Zhandry. Unclonable decryption keys. *Cryptology ePrint Archive*, 2020.

[JKKZ21] Ruta Jawale, Yael Tauman Kalai, Dakshita Khurana, and Rachel Yun Zhang. SNARGs for bounded depth computations and PPAD hardness from sub-exponential LWE. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 708–721, 2021.

[JKLV24] Zhengzhong Jin, Yael Kalai, Alex Lombardi, and Vinod Vaikuntanathan. SNARGs under LWE via propositional proofs. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 1750–1757, 2024.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 723–732, 1992.

[KLVW23] Yael Kalai, Alex Lombardi, Vinod Vaikuntanathan, and Daniel Wichs. Boosting batch arguments and ram delegation. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 1545–1552, 2023.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 669–684, 2013.

[KPY19a] Yael Tauman Kalai, Omer Paneth, and Lisa Yang. How to delegate computations publicly. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 1115–1124, 2019.

[KPY19b] Yael Tauman Kalai, Omer Paneth, and Lisa Yang. How to delegate computations publicly. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 1115–1124, 2019.

[LMS22] Alex Lombardi, Fermi Ma, and Nicholas Spooner. Post-quantum zero knowledge, revisited or: How to do quantum rewinding undetectably. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 851–859. IEEE, 2022.

[Mah18] Urmila Mahadev. Classical verification of quantum computations. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 259–267. IEEE, 2018.

[Mer87] Ralph C Merkle. A digital signature based on a conventional encryption function. In *Conference on the theory and application of cryptographic techniques*, pages 369–378. Springer, 1987.

[Mic00] Silvio Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000.

[MNZ24] Tony Metger, Anand Natarajan, and Tina Zhang. Succinct arguments for qma from standard assumptions via compiled nonlocal games, 2024.

[Nao03] Moni Naor. On cryptographic assumptions and challenges (invited talk). In *Annual International Cryptology Conference*, pages 96–109. Springer, 2003.

[NWW23] Shafik Nassar, Brent Waters, and David J Wu. Monotone policy bargs from bargs and additively homomorphic encryption. *Cryptology ePrint Archive*, 2023.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, pages 84–93, New York, NY, USA, 2005. ACM.

[RRR16] Omer Reingold, Guy N Rothblum, and Ron D Rothblum. Constant-round interactive proofs for delegating computation. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 49–62, 2016.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 475–484, 2014.

[Unr12] Dominique Unruh. Quantum proofs of knowledge. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 135–152. Springer, 2012.

[WW21] Hoeteck Wee and Daniel Wichs. Candidate obfuscation via oblivious lwe sampling. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 127–156. Springer, 2021.

[WW22] Brent Waters and David J Wu. Batch arguments for np and more from standard bilinear group assumptions. In *Annual International Cryptology Conference*, pages 433–463. Springer, 2022.

[Zha23] Mark Zhandry. Tracing quantum state distinguishers via backtracking. In *Annual International Cryptology Conference*, pages 3–36. Springer, 2023.

A Public Coin Succinct Arguments for QMA

In this section, we outline the barriers to achieving a six round succinct argument system for QMA by modifying the argument system presented in [BKL+22]. We choose this version of the protocol for ease of presentation and thus require a post-quantum indistinguishability obfuscation as an underlying assumption. However, the analysis for [GKNV24] and [MNZ24] follow similarly as they rely on AoK-style full extraction. The key idea is to reduce the number of interactions by interleaving the messages of the AoK system [CMSZ22] with the messages of the commitment procedure. We first outline a quantum knowledge extraction procedure for Classical Proof-of-Knowledge (PoK) systems to motivate our final construction. Finally, we present two succinct argument systems for QMA (eight and six rounds respectively) and note the barriers to the six-round construction.

A.1 Quantum Knowledge extraction from Classical PoK systems

Knowledge extraction from Σ-protocols - three round protocols between classical participants with uniform random messages from the verifier - can be done via rewinding a prover and extracting the witness from two accepting transcripts. Σ-protocols have two flavors: Proof-of-Knowledge and Argument-of-Knowledge. PoKs have statistical soundness guarantees while AoKs offer computational soundness. The rewinding procedure does not translate trivially to quantum provers as any intermediate measurement disturbs the quantum states used and thus we cannot obtain two accepting transcripts. Unruh's [Unr12] work addresses these concerns for Σ-protocols that satisfy the following special properties

Definition A.1 (**Special Soundness**)**.** A Σ-protocol (P,V) for a relation R has *special soundness* iff there exists a deterministic poly-time algorithm K_0 s.t: For any two accepting conversations $(com, ch, resp)$, and $(com, ch', resp')$ for x s.t $ch \neq ch'$ are two challenges from the challenge-space C, we have $w := \mathsf{K}_0(x, com, ch, resp, ch', resp')$ satisfies $(x, w) \in R$

Definition A.2 (**Strict Soundness**)**.** A Σ protocol (P,V) for a relation R has *strict soundness* iff for any two accepting conversations $(com, ch, resp)$ and $(com, ch, resp')$ for x, we must have $resp = resp'$.

The central result of the paper [Unr12] can be summarized by the following theorem below

Theorem A.3. Let (P,V) be a Σ-protocol for a relation R with challenge space C. Fix a function c s.t for all $\lambda \in \mathbb{N}, x \in \{0,1\}^*$, we have $|C(\lambda, x)| \geq c(\lambda)$. Assume that (P,V) has special soundness and strict soundness. Then (P,V) is quantum extractable with a knowledge error $1/\sqrt{c}$

The proof crucially relies on the following assumptions:

1. The strict soundness property must be statistical in nature, i.e there cannot be multiple accepting transcripts for the same challenge.
2. The cheating prover can be purified to defer all measurements to the end of the protocol and thus can be applied as a unitary operation on any initial state.
3. For a given challenge space, each challenge unitary must be orthogonal.

At a high-level, the proof generates two accepting transcripts uses the canonical extractor K_0 for the two challenges $ch, ch' \in C$. To ensure that the transcripts for ch, ch' are accepting, one of the

hybrids in the proof performs a measurement by purifying the cheating prover's operations and checking if the final result is acceptance. Strict soundness ensures that the post-measurement state is undisturbed after these measurements. Special soundness ensures that the extractor K_0 exists and is efficient.

Remark A.4. Unruh argues that it is unclear whether the extractor works for an Argument-of-Knowledge system with computational strict soundness. This would imply that any output measurement state for *resp* may be in a superposition of multiple eigenstates. Thus, any measurement via the purified cheating prover will disturb the post measurement state.

A.2 Succinct Argument systems for QMA

Version 1

1. Define the circuit $C : \{0,1\}^{\log N} \to \{0,1\}$ s.t $C(i) = h_i$ and C is obfuscated. The Verifier computes and sends $(\mathsf{ct}_r, \mathsf{mh.pk}, H)$ to the prover.

$$\begin{aligned} \mathsf{mh.pk} &\leftarrow \mathsf{Maha.Gen}(1^\lambda, C; r) \\ \mathsf{ct}_r &= \mathsf{FHE.Enc}(\mathsf{FHE.pk}, r) \\ H &\leftarrow \{\mathcal{H}_\lambda\}_{\lambda\in\mathbb{N}} \end{aligned}$$

   where $\{\mathcal{H}_\lambda\}_{\lambda\in\mathbb{N}}$ is a collapsing hash family (see [CMSZ22, Definition 3.9]).

2. The prover computes the proof state $|\psi\rangle$ using the FHM protocol and commits to it using $\mathsf{mh.pk}$. Let $(y, |\mathsf{st}\rangle)$ be the output of the $\mathsf{Maha.Com}$ procedure. The prover initializes the AoK for the relation $\mathcal{R}_\mathcal{G}$ (see Defintion 4.2) then computes $\hat{y} = H(y)$ and sends it to the prover.

3. The verifier samples a random question Q_1 for the AoK and sends them to the prover

4. The prover responds with the responses R_1 to the AoK

5. The verifier checks if the AoK responses R_1 are valid. Then it samples a random bit $c \in \{0,1\}$ and sends it to the prover.

6. The prover runs $z \leftarrow \mathsf{Maha.Open}(|\mathsf{st}\rangle, c)$ and computes $\hat{z} = H(z)$. Subsequently, the prover computes $\mathsf{ct}_P = \mathsf{FHE.Eval}(\mathsf{FHE.pk}, \mathsf{VerifyInstance}(x, y, z, c, \cdot), \mathsf{ct}_r)$ and sends $(\hat{z}, \mathsf{ct}_P)$ to the verifier.

7. The verifier runs an AoK check for the language

$$\begin{aligned} \exists w_1, w_2, \mathsf{ct}_P \text{ s.t } \hat{y} &= H(w_1) \wedge \\ \hat{z} &= H(w_2) \wedge \\ \mathsf{ct}_P &= \mathsf{FHE.Eval}(\mathsf{FHE.pk}, \mathsf{VerifyInstance}(x, w_1, w_2, c, \cdot), \mathsf{ct}_r) \end{aligned}$$

   So, the verifier samples a random question Q_2 for the AoK and sends it to the prover.

8. The prover responds with responses R_2 for the AoK.

9. The verifier checks if the AoK responses R_2 are valid and checks if ct_P decrypts to 1.

Version 2

1. The Verifier computes and sends $(\mathsf{ct}_r, \mathsf{mh.pk}, H)$ to the prover as done previously.
2. The prover computes $\hat{y} = H(y)$ and sends it to the prover as done previously.
3. The verifier samples a random question Q_1 for the AoK, a random bit c, and sends (c, Q_1).
4. The prover computes $\hat{z}, \mathsf{ct}_P$, and computes the responses to the AoK R_1 and sends $(\hat{z}, \mathsf{ct}_P, R_1)$
5. The verifier checks if the AoK responses R_1 are valid. Then, the verifier samples a random question Q_2 for the AoK and sends it to the prover.
6. The prover responds with responses R_2 for the AoK.
7. The verifier checks if the AoK responses R_2 are valid and checks if ct_P decrypts to 1.

The highlighted items in **Version 1** of the protocol are the additional two messages that are introduced by the AoK for the relation $\mathcal{R}_{\mathcal{G}}$. The highlighted items in **Version 2** of the protocol shows how the challenge bit c of the Maha protocol is interleaved with the AoK messages. At first glance, the **Version 2** of the protocol has six rounds as compared to eight rounds for **Version 1**. Moreover, the completeness of both protocols follow from the completeness properties of the primitives used in the construction. For the soundness proof for **Version 2** of the protocol consider a reduction algorithm $\mathcal{B}$ for a cheating QPT prover $\mathcal{A}$ as below.

1. A challenger $\mathcal{C}$ to the $\mathsf{VerifyInstance}$ predicate samples the $(\mathsf{mh.pk}, \mathsf{ct}_r)$ as outlined above and sends it to the reduction algorithm $\mathcal{B}$.
2. The reduction algorithm receives $(\mathsf{mh.pk}, \mathsf{ct}_r)$ and samples a random collapsing hash function H. Finally, it sends $(\mathsf{mh.pk}, \mathsf{ct}_r, H)$ to the cheating prover $\mathcal{A}$.
3. The prover computes the message $\hat{y}$ and sends it to the reduction algorithm $\mathcal{B}$.
4. The reduction algorithm $\mathcal{B}$ runs the AoK extractor to extract the output y from the prover $\mathcal{A}$ with a random bit $c' \in \{0, 1\}$ and forwards y to the challenger $\mathcal{C}$.
5. The challenger samples a random bit $c \in \{0, 1\}$ and sends it to $\mathcal{B}$ which then forwards this to the prover $\mathcal{A}$.
6. The prover computes the message $\hat{z}$ and sends it to the reduction algorithm $\mathcal{B}$.
7. The reduction algorithm runs the AoK extractor to extract the output (y', z) from the prover $\mathcal{A}$ and sends z to the challenger $\mathcal{C}$.

The reduction algorithm ensures that $y = y'$ and if $\mathcal{A}$ succeeds in convincing the verifier of the **Version 2** of the protocol, we have a reduction algorithm that breaks the soundness guarantees of the $\mathsf{VerifyInstance}$ predicate. However, a few subtleties make the above proof strategy infeasible. Firstly, the knowledge-system used in the version of the protocol above is an Argument-of-Knowledge and not a Proof-of-Knowledge and thus has computational soundness. Therefore, we cannot use Unruh's result as a black-box.

To attempt a rewinding proof as in Theorem A.3, we can allow the reduction algorithm $\mathcal{B}$ access to operate separately on the registers $\mathcal{Y}, \mathcal{Z}$ of the cheating prover for the Commit-and-Measure protocol based on specific control qubits. Let $c^\star \in \{0,1\}$ be the challenge sent by the reduction algorithm $\mathcal{B}$ in round 3 of the protocol. Let $\Pi_{c^\star}$ be the projector that measures whether the prover gets accepted for challenge $c^\star$ and outputs the value of y that gets accepted. In essence, we run the cheating prover $\mathcal{A}$ with input $c^\star$ and measure if we get an accepting transcript and only then measuring $\mathcal{Y}$ in the computational basis. We proceed as in Theorem A.3 and extract y, which can be then forwarded to the challenger $\mathcal{C}$.

For the proof strategy to work, we would require $\Pi_{c^\star}$ to not significantly disturb the registers $\mathcal{Y}$ and $\mathcal{Z}$, i.e we need it to obey strict soundness. It is unclear, however, that the Mahadev measurement protocol satisfies such statistical guarantees. The soundness guarantees for the protocol are computational in nature and thus any measurement by the projector $\Pi_{c^\star}$ can potentially disturb the opening register $\mathcal{Z}$. We do not know (to the best of our knowledge) any ways to navigate this issue other than using **Version 1** of the protocol.

B Proof of Lemma 7.23

Lemma B.1. Assuming the index hiding and extraction correctness properties of seBARG_2, extraction correctness, extraction consistency, and consistency of tag extraction properties of PEHT_2 and the predicate hiding property of PEHash, for any QPT attacker $\mathcal{A}$, the attacker's advantage is indistinguishable in **Hybrid 7** and **Hybrid 8**

Proof. Recall the current state of the hybrids. We have

1. Enabled extraction for PEHT_1 on the non uniform advice of the necessary set $J^\star$.
2. Ensured that the string s used for h_y and h_z is consistent across **Round 1** and **Round 3**.
3. Enabled the extraction for PEHT_2 on $J^\star$ and checked if $j_3 \in J^\star$

Let us define a relation $\mathcal{R}_t$ s.t $\mathcal{R}_t(x_j, w_j) = 1$ with $\mathsf{op}_j^{y\star}, \mathsf{op}_j^{z\star}, y_j^{\star\prime}, z_j^{\star\prime}, s_j^\star) = w_j$ iff the following checks are true

$$\begin{aligned}
\mathsf{PEHT}_1.\mathsf{Verify}(\mathsf{pT}.\mathsf{td}_1^\star, \mathsf{h}_y, j, s_j^\star, \mathsf{op}_j^{y\star}) = 1 \wedge \\
\mathsf{PEHT}_1.\mathsf{Verify}(\mathsf{pT}.\mathsf{td}_1^\star, \mathsf{h}_z, j, s_j^\star, \mathsf{op}_j^{z\star}) = 1 \wedge \\
\mathsf{VerifyInstance}(x_j, y_j, z_j, c, r) = 1 \ \wedge \ s_j^\star = 1
\end{aligned}$$

We say that attacker $\mathcal{A}$ wins with non-negligible, say $\mu_6(\lambda)$, advantage across **Hybrid 7** and **Hybrid 8** if he can output $\mathcal{R}_t(x_{j_3}, w_{j_3}) = 0$ and still get accepted by the challenger $\mathcal{C}$. We are essentially proving somewhere argument of knowledge for Monotone Policy Batch SNARGs with key ideas coming from the proof of [BBK+23, Theorem. 7.1]. Let d be the depth of the circuit C_f. For all $i \in [d]$ denote $J_i \subset [M]$ the wires in the ith layer (excluding the input layer) that have values 0. We now define a sequence of intermediate hybrids in which we update the Setup algorithm to enable extraction for PEHash. Let the intermediate set of hybrids be denoted by $\left(\mathsf{Hyb4}_{i,j}\right)_{0 \le i,j \le d}$ where we make the following updates

$\mathsf{Setup}_{i,j}(1^k, 1^k, 1^N, f)$ does the following:

(a) Define predicates $d_{f_1} = (b, i)$ and $d_{f_2} = (b, j)$, where $b \in \{0,1\}^k$ is defined by $b_i = \mathbf{1}_{i \notin J^\star}$.
(b) Generate $(\mathsf{pH.hk}^\star_\beta, \mathsf{pH.vk}^\star_\beta, \mathsf{pH.td}^\star_\beta) \leftarrow \mathsf{PEHash.Gen}(1^\lambda, M, d_{f_\beta})$ for $\beta \in [2]$ and $M = |C_f|$.

We have the following claim

Claim B.2. Assuming the predicate hiding property of PEHash, the QPT attacker $\mathcal{A}$ must have an advantage that is indistinguishable between $\mathsf{Hyb4}_{i,i}$ and $\mathsf{Hyb4}_{i,i-1}$ for all $i \in [d]$.

Proof. This follows from the proof of Lemma 7.11. Therefore, there exists a negligible function $\mathsf{negl}(\cdot)$ s.t. the advantage of $\mathcal{A}$ in $\mathsf{Hyb4}_{i,i-1}$ is $\mu_4(\lambda) - \mathsf{negl}(\lambda)$ □

For every $i \in [d]$ and for every $\lambda \in \mathbb{N}$, denote

$$\nu_i(\lambda) = \nu_i^{(1)}(\lambda) \triangleq \Pr_{\mathsf{Hyb4}_{i,i-1}}\left[\begin{array}{l}\mathcal{R}_t(x_{j_3}, w_{j_3}) = 0 \wedge \\ \mathsf{PEHash.Extract}(\mathsf{pH.td}^\star_1, \mathsf{v}_1, C_f) = 0\end{array}\right],$$

$$\nu_{i-1}^{(2)} \triangleq \Pr_{\mathsf{Hyb4}_{i,i-1}}\left[\begin{array}{l}\mathcal{R}_t(x_{j_3}, w_{j_3}) = 0 \wedge \\ \mathsf{PEHash.Extract}(\mathsf{pH.td}^\star_2, \mathsf{v}_2, C_f) = 0\end{array}\right],$$

$$\nu_{\mathsf{pT}}(\lambda) \triangleq \Pr_{\mathsf{Hyb4}_{0,0}}\left[\begin{array}{l}\mathcal{R}_t(x_{j_3}, w_{j_3}) = 0 \wedge \\ \mathsf{PEHT}_2\mathsf{.Extract}(\mathsf{pT.td}^\star_2, \mathsf{h}_w) = 0\end{array}\right]$$

Claim B.3. Assuming the index hiding and somewhere argument of knowledge property of seBARG_2, and extraction consistency of PEHash, for every QPT attacker $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ s.t. for all $\lambda \in \mathbb{N}$ and $i > 1$

$$\nu_i^{(1)}(\lambda) \leq \nu_{i-1}^{(2)}(\lambda) + \mathsf{negl}(\lambda)$$

Proof. By simple probability arguments, we have

$$\nu_i^{(1)}(\lambda) \leq \nu_i^{(2)}(\lambda) + \Pr_{\mathsf{Hyb4}_{i,i-1}}\left[\begin{array}{l}j_3 \notin J^\star \ \vee \ \mathcal{R}_t(x_{j_3}, w_{j_3}) = 0 \wedge \\ \mathsf{PEHash.Extract}(\mathsf{pH.td}^\star_1, \mathsf{v}_1, C_f) = 0 \wedge \\ \mathsf{PEHash.Extract}(\mathsf{pT.td}^\star_2, \mathsf{v}_2, C_f) = 1\end{array}\right] \leq \nu_i^{(2)}(\lambda) + \delta_{i,i-1}(\lambda)$$

For contradiction, assume $\delta_{i,i-1}(\lambda)$ is non-negligible. Let the attacker $\mathcal{A}$ win if it triggers the above condition. We have the following for any index j' in the ith level of C_f.

$$\Pr\left[\mathcal{A} \text{ wins} \wedge j' = \mathsf{ExtractIndex}(\mathsf{pH.td}^\star_1, \mathsf{v}_1)\right] = \frac{\delta_{i,i-1}(\lambda)}{\mathrm{width}(C_f)} \geq \frac{\delta_{i,i-1}(\lambda)}{\mathsf{poly}(\lambda)} = \delta'_{i,i-1}(\lambda)$$

The challenger enables extraction on the set $\{j' = j'_0, j'_1, j'_2\}$ for seBARG_2 where j'_1, j'_2 are the inputs to the gate $G_{j'}$

$$(\mathsf{sB.crs}^\star_2, \mathsf{sB.td}^\star_2) \leftarrow \mathsf{seBARG}_2\mathsf{.Setup}(1^\lambda, 1^M, 1^T, \{j', j'_1, j'_2\})$$

We characterize the attacker $\mathcal{A}$ into different types of attackers. Let $\mathcal{A}_1$ be an attacker for the index hiding property of seBARG_2.

Claim B.4. Assuming index hiding property of seBARG_2, the QPT attacker $\mathcal{A}_1$ must have an advantage that is indistinguishable from the advantage of $\mathcal{A}$

Proof. This follows from the proof of Lemma 4.7. Therefore, there exists a negligible function $\mathsf{negl}(\cdot)$ s.t. the advantage of $\mathcal{A}$ is $\delta'_{i,i-1}(\lambda) - \mathsf{negl}(\lambda)$ □

From here, we extract a witness at the location j' using seBARG extraction, i.e.,

$$\omega^\star_{j'} = \left(b^{\star\prime}_{j'}, b^{\star\prime}_{j'_1}, b^{\star\prime}_{j'_2}, \left(\rho^{\star(\beta)}_{j'}, \rho^{\star(\beta)}_{j'_1}, \rho^{\star(\beta)}_{j'_2}\right)_{\beta\in[2]}\right)$$

We may also get $\omega^\star_{j'_1}$ and $\omega^\star_{j'_2}$ as potential witnesses. Let $\mathcal{A}_2$ be an attacker for the somewhere argument of knowledge property for seBARG_2, while $\mathcal{A}_3$ an attacker for the consistency of extraction property of PEHT_2. We now have the following claims.

Claim B.5. Assuming the somewhere argument of seBARG_2 and computational binding of PEHash, the QPT attacker $\mathcal{A}_2$ must have an advantage that is indistinguishable from $\mathcal{A}$.

Proof. This follows from the proof of Lemma 4.8. Thus, we have $\chi(\zeta, j'_\alpha, \omega^\star_{j'_\alpha}) = 1$ for $\alpha \in \{0,1,2\}$. Therefore, there exists a negligible function $\mathsf{negl}(\cdot)$ s.t. the advantage of $\mathcal{A}$ is $\delta'_{i,i-1}(\lambda) - \mathsf{negl}(\lambda)$ □

Claim B.6. Assuming the extraction correctness, consistency of extraction, of PEHash, the QPT attacker $\mathcal{A}_3$ must have an advantage that is indistinguishable from the advantage of $\mathcal{A}$.

Proof. Consider a reduction algorithm $\mathcal{B}$ that interacts with a challenger $\mathcal{C}$ for the consistency of extraction property. Suppose that we have $b^{\star\prime}_{j'} = 1 = \bar{m}_{j'}$. Since $j' = \mathsf{PEHash.ExtractIndex}(\mathsf{pH.td}^\star_1, \mathsf{v}_1) \in J^\star$, we must have

$$\mathsf{PEHash.Extract}(\mathsf{pT.td}^\star_1, \mathsf{v}_1) = 0$$

otherwise $\mathcal{B}$ trivially uses $\mathcal{A}_3$ to produce $(\mathsf{v}_1, j', b^{\star\prime}_{j'}, \rho^\star_{j'})$ and break the consistency of extraction property of PEHash. Therefore, there exists a negligible function $\mathsf{negl}(\cdot)$ s.t. the advantage of $\mathcal{A}$ is $\delta'_{i,i-1}(\lambda) - \mathsf{negl}(\lambda)$. □

At this point, we have shown that Gate $G_{j'}$ has the right output value. However, since $b^\star_{j'} = 0$ (by assumption), one of the child nodes must be evaluated incorrectly. We now have the following for any random value of $\alpha \in \{1,2\}$

$$\begin{aligned}\mathsf{Pr}\left[\mathcal{A} \text{ wins} \wedge j' = \mathsf{PEHash.ExtractIndex}(\mathsf{pH.td}^\star_1, \mathsf{v}_1) \wedge b^{\star\prime}_{j'_\alpha} > b^\star_{j'_\alpha}\right] &\geq \frac{\delta'_{i,i-1}(\lambda)}{2} - \mathsf{negl}(\lambda)\\ &= \delta''_{i,i-1}(\lambda)\end{aligned}$$

Since the actual value on wire $b^\star_{j'}$ and the extracted value $b^{\star\prime}_{j'}$ don't match, we must have the above probability equation to hold for at least one of the input wires to gate $G_{j'}$. Without loss of generality, let that input wire be j'_1. We have the following claim when $j'_1 > k$

Claim B.7. Assuming the extraction consistency of PEHash, for any QPT attacker $\mathcal{A}$, the advantage of the attacker is negligible for all $i > 1$

Proof. This follows from Claim B.6. Therefore, we must have the advantage of the attacker $\mathcal{A}$, $\delta''_{i,i-1}(\lambda)$, be negligible. □

Since $\delta''_{i,i-1}(\lambda) = \mathsf{negl}(\lambda)$, we must have $\delta'_{i,i-1}(\lambda) = \delta_{i,i-1}(\lambda) = \mathsf{negl}(\lambda)$. □

At this point in the proof we have shown that if layer i is computed correctly, then layer $i-1$ must also be computed correctly. However, we are yet to move from layer i to $i-1$ for the first instance of PEHash. For this, we make use of the following claim.

Claim B.8. Assuming the index hiding and somewhere argument of knowledge property of seBARG_2, and extraction consistency of PEHash, for every QPT attacker $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ s.t, for all $\lambda \in \mathbb{N}$

$$\nu^{(2)}_{i-1}(\lambda) \leq \nu^{(1)}_{i-1}(\lambda) + \mathsf{negl}(\lambda)$$

Proof. By predicate hiding property of PEHash (Claim B.2), we have

$$\nu^{(2)}_{i-1}(\lambda) \leq \Pr_{\mathsf{Hyb4}_{i-1,i-1}}\left[\mathsf{PEHash.Extract}(\mathsf{pH.td}^\star_2, \mathsf{v}_2) = 0\right] + \mathsf{negl}(\lambda)$$

We have by simple probability arguments

$$\nu^{(2)}_{i-1}(\lambda) \leq \Pr_{\mathsf{Hyb4}_{i-1,i-1}}\left[\mathsf{PEHash.Extract}(\mathsf{pH.td}^\star_1, \mathsf{v}_1) = 0\right] + \Pr_{\mathsf{Hyb4}_{i-1,i-1}}\left[\begin{array}{l}\mathsf{PEHash.Extract}(\mathsf{pH.td}^\star_2, \mathsf{v}_2) = 0 \wedge \\ \mathsf{PEHash.Extract}(\mathsf{pH.td}^\star_1, \mathsf{v}_1) = 1\end{array}\right] + \mathsf{negl}(\lambda)$$

The reason why we can omit the check $\mathsf{PEHash.Extract}(\mathsf{pH.td}^\star_2, \mathsf{v}_2) = 0$ in the first term is because they represent the same event. Moreover, we would want to drop it in order to establish the following inequality by using the predicate hiding property (by Claim B.2)

$$\Pr_{\mathsf{Hyb4}_{i-1,i-1}}\left[\mathsf{PEHash.Extract}(\mathsf{pH.td}^\star_1, \mathsf{v}_1) = 0\right] \leq \nu^{(1)}_{i-1} + \mathsf{negl}(\lambda)$$

Thus, we finally have

$$\nu^{(2)}_{i-1}(\lambda) \leq \nu^{(1)}_{i-1} + \delta_{i-1}(\lambda) + \mathsf{negl}(\lambda)$$

where $\delta_{i-1}(\lambda)$ is defined as the probability that the first instance of PEHash extracts a value 1 on layer $i-1$, while the second instance of PEHash extracts the value of 0. We prove that δ_{i-1} is negligible for all $i > 1$ by extracting on a random index $j^\star$ in the $(i-1)^{\text{th}}$ layer of C_f, i.e.,

$$(\mathsf{sB.crs}^\star_2, \mathsf{sB.td}^\star_2) \leftarrow \mathsf{seBARG}_2.\mathsf{Setup}(1^\lambda, 1^M, 1^T, \{1, 2, j^\star\})$$

The proof for the same proceeds as done Claim B.3. Thus, we must have $\delta_{i-1}(\lambda) = \mathsf{negl}(\lambda)$ □

At this point, we have shown that $\nu_d(\lambda) \leq \nu^{(2)}_0(\lambda) + \mathsf{negl}(\lambda)$. We are yet to show that the input layer is consistent with the output. First, we show that the $\mathsf{Extract}$ output of PEHT_2 is consistent with the output. We prove it through the following claim.

Claim B.9. Assuming the extraction correctness of PEHT, for any QPT attacker $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ s.t for all $\lambda \in \mathbb{N}$

$$\nu_0(\lambda) = \nu^{(2)}_0(\lambda) \leq \nu_{\mathsf{pT}}(\lambda) + \mathsf{negl}(\lambda)$$

Proof. From the steps of Claim B.8, we have the following

$$\nu_0^{(2)}(\lambda) \leq \nu_0^{(1)}(\lambda) + \mathsf{negl}(\lambda)$$

Thus, we have

$$\nu_0(\lambda) \leq \nu_{\mathsf{pT}}(\lambda) + \Pr_{\mathsf{Hyb4}_{0,0}}\left[\begin{array}{l} j_3 \notin J^\star \ \vee \ \mathcal{R}_t(x_{j_3}, w_{j_3}) = 0 \ \wedge \\ \mathsf{PEHash.Extract}(\mathsf{pH.td}_2^\star, \mathsf{v}_2, C_f) = 0 \ \wedge \\ \mathsf{PEHT}_2.\mathsf{Extract}(\mathsf{pT.td}_2^\star, \mathsf{h}_w) = 1 \end{array}\right] \leq \nu_{\mathsf{pT}}(\lambda) + \delta(\lambda)$$

Assume for contradiction that $\delta(\lambda)$ is non-negligible. By definition, we have both v_2 and h_w extractable on $(J^\star, 0^{J^\star})$. Thus, by extraction correctness, we must have $\delta(\lambda) = \mathsf{negl}(\lambda)$ □

Finally, we show that the witness extracted through $\mathsf{ExtractTag}$ of PEHT_2 is consistent with the witness extracted by the seBARG_2.

Claim B.10. Assuming the index hiding and somewhere argument of knowledge of seBARG_2, the consistency of extraction properties of PEHT_2, for any QPT attacker $\mathcal{A}$, we must have $\nu_{\mathsf{pT}}(\lambda)$ as a negligible function for all $\lambda \in \mathbb{N}$.

Proof. We have the following for any index $j^\star \in [k]$:

$$\Pr[\mathcal{A} \text{ wins } \wedge \ j^\star = j_3] = \frac{\nu_{\mathsf{pT}}(\lambda)}{k} = \frac{\nu_{\mathsf{pT}}(\lambda)}{\mathsf{poly}(\lambda)} = \nu'_{\mathsf{pT}}(\lambda)$$

The challenger enables extraction on the set $\{1, 2, j^\star\}$ for seBARG_2.

$$(\mathsf{sB.crs}_2^\star, \mathsf{sB.td}_2^\star) \leftarrow \mathsf{seBARG}_2.\mathsf{Setup}(1^\lambda, 1^M, 1^T, \{1, 2, j^\star\})$$

From here, we follow the arguments of Claim B.3 and assert that $\nu_{\mathsf{pT}}(\lambda)$ must be negligible. □

Thus we have $\nu_0^{(2)}(\lambda) = \mathsf{negl}(\lambda)$, thereby implying that $\nu_d(\lambda) = \mu_6(\lambda) = \mathsf{negl}(\lambda)$, a contradiction. Therefore, there exists a negligible function $\mu_6(\lambda)$ s.t the advantage of attacker $\mathcal{A}$ in **Hybrid 8** is $\varepsilon(\lambda) - \mu_1(\lambda) - \mu_2(\lambda) - \mu_3(\lambda) - \mu_4(\lambda) - \mu_5(\lambda) - \mu_6(\lambda)$. □